НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра автоматизованих систем обробки інформації і управління

«На правах рукопису»

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)

УДК _____

"___" _____2019 р.

Магістерська дисертація

зі спеціальності *121 «Інженерія програмного забезпечення»*

на тему: «*Інтелектуальна система для аналізу кредитного портфеля банка*»

Виконав: студент VI курсу, групи *ІП-82мп*

Романовський Юрій Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник *доц., доц., к.т.н. Селін Ю.М,* _____
(посада, науковий ступінь, , вчене звання, прізвище та ініціали) (підпис)

Консультант *доц., к.т.н., Ліщук К.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *старший науковий співробітник ІПСА, к.т.н. Кондратова Л.П.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти *другий (магістерський) за освітньо-професійною програмою*

Спеціальність *121 «Інженерія програмного забезпечення»*
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
«___» _____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Романовському Юрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації *Інтелектуальна система для аналізу кредитного портфеля банка*

науковий керівник дисертації *к.т.н., доц. Селін Ю.М.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "____" _____20__ р. № _____

2. Строк подання студентом дисертації "____" _____20__ р.

3. Об'єкт дослідження *Аналіз кредитного портфелю банку*

4. Предмет дослідження *Методи аналізу кредитного портфелю банку та модель ефективного управління ним*

5. Перелік завдань, які потрібно розробити *Огляд підходів та методів кредитного портфелю, розробка формальної постановки задачі, моделювання*

Та визначення оптимальної моделі, розробка програмної реалізації інтелектуальної системи аналізу кредитного портфеля.

6. Перелік графічного матеріалу

Схема структурна классів модулю Execution

Діаграма сутностей

7. Орієнтовний перелік публікацій *Інформаційна система підтримки прийняття рішень в управлінні кредитною заборгованістю, Аналіз кредитного портфеля банка*

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання “ *01* ” *вересня* 20 *19* р

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Огляд теоретичних та методологічних основ оцінки якості кредитного портфеля*	*23.09.2019*	
2	*Огляд основних підходів та методів аналізу кредитного портфеля*	*02.10.2019*	
3	*Постановка та формалізація математичної моделі задачі*	*06.10.2019*	
4	*Аналіз результатів моделювання та оцінювання задля обґрунтованого вибору найкращої моделі*	*18.10.2019*	
5	*Розробка інформаційного та програмного забезпечення*	*24.10.2019*	
7	*Проведення експериментальних досліджень розробленого алгоритму*	*09.11.2019*	
8	*Оформлення документації*	*15.11.2019*	
9	*Подання роботи на попередній захист*	*05.12.2019*	
10	*Подання роботи на основний захист*	*16.12.2019*	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність. Актуальність теми в сучасних умовах пояснюється, перш за все, ризикованістю кредитного портфеля комерційного банку при його формуванні, а також необхідністю своєчасної та повної оцінки стану як окремо взятого, так і сукупності всіх кредитів. Успішне здійснення кредитних операцій призводить до отримання банками прибутку, що сприяє підвищенню надійності і стійкості кредитної організації. Також банківського кредиту притаманне важлива перевага, що полягає в гнучкому задоволенні мінливих потреб позичальників в засобах. Таким чином, у розвитку системи банківського кредитування зацікавлені як самі банки, так і позичальники.

Метою дослідження є побудова системи підтримки прийняття рішень для ефективного управління кредитним портфелем.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- вивчити теоретичні та методологічні основи оцінки якості кредитного портфеля;
- провести огляд основних підходів;
- здійснити моделювання;
- проаналізувати результати моделювання та оцінювання задля обґрунтованого вибору найкращої моделі;
- виконати програмну реалізацію системи підтримки прийняття рішень з використанням обраної моделі;
- провести дослідження ефективності розробленої інформаційної технології.

Об’єктом дослідження є аналіз кредитного портфелю банку.

Предметом дослідження є методи аналізу кредитного портфелю банку та модель ефективного управління ним.

Методами дослідження є методи машинного навчання, які базуються на алгоритмі стохастичного градієнтного спуску.

Наукова новизна. Створено систему підтримки прийняття рішень для ефективного управління кредитним портфелем, що базується на алгоритмах машинного навчання та здатна класифікувати кредити на основі значної кількості важливих параметрів позики та клієнта, яка дозволяє значно підвищити якість аналізу та ефективність управління кредитною заборгованністю.

Розроблена система є універсальною і може використовуватися будь-яким банком та надає можливість користувачу вибирати за якими критеріями проводити класифікацію.

Практичне значення отриманих результатів визначається тим, що запропонований алгоритм дійсно є ефективним для класифікації банківських позичок, а розроблена система підтримки прийняття рішень значно підвищує якість аналізу кредитного портфеля.

Публікації. За напрямком дослідження було опубліковано такі матеріли:

1) Романовский Ю. А. Информационная система поддержки принятия решений в управлении кредитной задолженностью // Романовский Ю. А., Селин Ю.Н./ Материали за XV международна научна практична конференция, Образованието и науката на XXI век - 2019 , 15 - 22 октомври 2019 г. Закон. Икономики. Публичната администрация. Философия. : София. « Бял ГРАД-БГ » - с.6-9.

2) Романовский Ю. А. Анализ кредитного портфеля банка // Романовский Ю. А., Селин Ю.Н./ Materiałły XV Międzynarodowej naukowi-praktycznej konferencji , «Perspektywiczne opracowania są nauką i technikami - 2019» , 07 – 15 listopada 2019 roku po sekcjach: Ekonomiczne nauki.- с.34-38.

Ключові слова: КРЕДИТНИЙ ПОРТФЕЛЬ, МАШИНЕ НАВЧАННЯ, КРЕДИТНА ЗАБОРГОВАННІСТЬ, СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ, СТОХАСТИЧНИЙ ГРАДІЄНТНИЙ СПУСК.

ABSTRACT

Actuality. The relevance of the topic in the current conditions is due, first of all, to the riskiness of the commercial bank's loan portfolio in its formation, as well as the need for timely and complete assessment of the status of both individual and all loans. Successful lending leads to a profit for banks, which enhances the credibility and sustainability of the lending institution. Also, a bank loan has the important advantage of being able to flexibly meet the borrower's changing needs for funds. Thus, both banks and borrowers are interested in the development of the bank lending system.

The purpose of the research is impulse system and pay-offs, we take the decision for the effective management of the loan portfolio.

To achieve this goal, you must perform the following tasks:

- to study theoretical and methodological bases of credit portfolio quality assessment;
- review the main approaches;
- to carry out modeling;
- analyze the results of modeling and evaluation to make a reasonable choice of the best model;
- implement software implementation of the decision support system using the selected model;
- to carry out research of efficiency of the developed information technology.

The object of the research is analysis of the loan portfolio of the bank.

The subject of the research is to analyze the loan portfolio of the bank and the model of effective management.

Methods of research are machine learning methods based on the stochastic gradient descent algorithm.

The scientific novelty. A decision support system for efficient credit portfolio management based on machine learning algorithms and capable of classifying loans based on a large number of important loan and customer parameters is established, which significantly improves the quality of analysis and efficiency of credit management.

The developed system is universal and can be used by any bank and allows the user to choose by what criteria to classify.

The practical significance of the obtained results is determined by the fact that the proposed algorithm is really effective for the classification of bank loans, and the developed system significantly improves the quality of credit portfolio analysis.

Publications. The following materials were published in the direction of the study:

1) Романовский Ю. А. Информационная система поддержки принятия решений в управлении кредитной задолженностью // Романовский Ю. А., Селин Ю.Н./ Материали за XV международна научна практична конференция, Образованието и науката на XXI век - 2019 , 15 - 22 октомври 2019 г. Закон. Икономики. Публичната администрация. Философия. : София. « Бял ГРАД-БГ » - с.6-9.

2) Романовский Ю. А. Анализ кредитного портфеля банка // Романовский Ю. А., Селин Ю.Н./ Materiały XV Międzynarodowej naukowi-praktycznej konferencji , «Perspektywiczne opracowania są nauką i technikami - 2019» , 07 – 15 listopada 2019 roku po sekcjach: Ekonomiczne nauki.- с.34-38.

Keywords: CREDIT PORTFOLIO, MACHINE LEARNING, CREDIT DEBT, DECISION MAKING SYSTEMS, STOCHASTIC GRADIENT.

ЗМІСТ

ВСТУП

У сучасних умовах розвитку діяльність кредитних організацій в першу чергу спрямована на досягнення конкурентних переваг; зміцнення своїх позицій на фінансовому ринку; забезпечення зростання вартості банківського бізнесу і прибутковості операцій, що проводяться.

Основними шляхами досягнення кредитними організаціями таких цілей є: розробка власної ніші в банківському бізнесі; нарощування обсягів операцій; освоєння нових ринків і розширення сфер ведення бізнесу; зниження витрат; оптимізація доходів і витрат.

Реалізація банком обраної стратегії розвитку завжди пов'язана з численними ризиками, оскільки банк у силу своєї діяльності є сполучною ланкою у фінансовій системі країни. При цьому однією з умов забезпечення стабільності банківської системи є ефективний банківський нагляд і високий рівень культури ведення бізнесу комерційних банків.

В даний час, в рамках створення банками систем управління ризиками, особливої актуальності набула проблема адекватної оцінки кредитного ризику - ризику невиконання клієнтом / контрагентом банку своїх кредитних зобов'язань перед ним. Пріоритетним напрямком в системі оцінки банківських ризиків і, зокрема, кредитного ризику є перевірка якості кредитного портфеля банку і кредитної політики, які, в свою чергу, відображають рівень і якість керівництва банком.

Необхідність проведення банками аналізу якості кредитного портфеля і контролю за ним обумовлена, головним чином, зміщенням банківських пріоритетів в бік якісного аналізу видаваних кредитів і розвитку систем управління ризиками.

Кредитні операції комерційних банків є найважливішим і найбільш динамічно розвиваються видом банківської діяльності, в силу їх найбільшої прибутковості в порівнянні з іншими видами активних операцій.

1 СУЧАСНИЙ СТАН АНАЛІЗУ КРЕДИТНОГО ПОРТФЕЛЯ

1.1 Сутність і поняття кредитного портфеля банку

Кредитна діяльність - один з найважливіших, що конституюють саме поняття банку. Рівень організації кредитного процесу - чи не найкращий показник всієї взагалі роботи банку і якості його менеджменту. Перш ніж почати видавати кредити, банк повинен сформулювати свою кредитну політику (поряд і в згоді з його політиками стосовно всіх інших напрямках діяльності - депозитної, процентної, тарифної, технічно, кадрової, по відношенню до клієнтури, до конкурентів і т.д.), а також передбачити способи і засоби її втілення в реальну практику. Формулювання політики (політик) банку становить один з етапів планування його діяльності. Визначити і затвердити свою кредитну політику - означає сформулювати і закріпити в необхідних внутрішніх документах позицію керівництва банку.

Для прийняття банком обгрунтованих рішень за вказаною кола питань важливе значення мають чітка і зважена постановка загальних цілей діяльності банку на майбутній період (тобто хороша постановка планування в цілому), адекватний аналіз кредитного ринку (тобто хороша робота маркетингової служби), ясність перспектив розвитку ресурсної бази банку, вірна оцінка якості кредитного портфеля, облік динаміки рівня кваліфікації персоналу та інші фактори. [2]

Всі положення кредитної політики спрямовані на те, щоб домогтися максимально можливої якості кредитної діяльності банку. Про якість кредитної діяльності банку (як організації банком своєї кредитної діяльності) можна судити по ряду критеріїв (ознак), серед яких:

- рентабельність кредитних операцій (в динаміці);
- наявність ясно сформульованої кредитної політики на кожен конкретний період, адекватної можливостям самого банку і інтересам його

клієнтів, а також чітко прописаних механізмів (включаючи організаційне та інформаційно-аналітичне забезпечення) і процедур реалізації такої політики (регламентів проведення всіх етапів кредитної операції);

- дотримання законодавства та нормативних актів, що відносяться до кредитного процесу;
- стан кредитного портфеля;
- наявність працюючого механізму управління кредитними ризиками. [1]

Кредитний портфель являє собою залишок кредитної заборгованості за балансом комерційного банку на певну дату. В економічній літературі кредитний портфель визначається як сукупність вимог банку по кредитах, які класифіковані на основі певних критеріїв. Одним з таких критеріїв, застосовуваних у закордонній та вітчизняній практиці, є ступінь кредитного ризику. За цим критерієм визначається якість кредитного портфеля. Аналіз і оцінка якості кредитного портфеля дозволяє менеджерам банку управляти його позичковими операціями. [2]

Аналіз кредитного портфеля банку проводиться регулярно і лежить в основі його управління, яке має на меті зниження сукупного кредитного ризику за рахунок диверсифікації кредитних вкладень і виявлення найбільш ризикових сегментів кредитного ринку. Основні етапи аналізу:

- визначення основних груп позичок із зазначенням пов'язаних з ними відсотків ризику;
- оцінка кожної виданої позики виходячи з обраних критеріїв, тобто віднесення її до відповідної групи;
- визначення структури кредитного портфеля в розрізі класифікованих позик;
- оцінка якості кредитного портфеля в цілому;
- аналіз чинників, що впливають на зміну структури кредитного портфеля в динаміці;

- визначення суми резервного фонду, адекватного сукупного ризику кредитного портфеля банку;
- розробка заходів щодо поліпшення якості кредитного портфеля. [1]

При формуванні «кредитного портфеля» необхідно враховувати такі ризики: кредитний, ліквідності і процентний.

Фактори кредитного ризику є основними критеріями його класифікації. Залежно від сфери дії факторів виділяються внутрішні і зовнішні кредитні ризики; від ступеня зв'язку факторів з діяльністю банку - кредитний ризик, залежний або незалежний від діяльності банку. Кредитні ризики, залежні від діяльності банку, з урахуванням її масштабів діляться на фундаментальні (пов'язані з прийняттям рішень менеджерами, що займаються управлінням активними і пасивними операціями); комерційні (пов'язані з напрямком діяльності ЦФО); індивідуальні та сукупні (ризик кредитного портфеля, ризик сукупності операцій кредитного характеру).

До фундаментальних кредитних ризиків відносяться ризики, пов'язані зі стандартами маржі застави, прийняттям рішень про видачу позик позичальникам, що не відповідає стандартам банку, а також є наслідком процентного і валютного ризику банку і т.д.

Комерційні ризики пов'язані з кредитною політикою щодо малого бізнесу, великих і середніх клієнтів - юридичних і фізичних осіб, з окремими напрямками кредитної діяльності банку.

Індивідуальні кредитні ризики включають ризик кредитного продукту, послуги, операції (угоди), а також ризик позичальника або іншого контрагента.

Для ризику ліквідності факторна сторона укладена в можливості не виконати зобов'язання перед вкладниками і кредиторами через відсутність необхідних джерел або виконати їх з втратою для себе.

До внутрішніх факторів ризику ліквідності прийнято відносити: якість активів і пасивів, ступінь незбалансованості активів і пасивів за строками,

сумами і в розрізі окремих валют, рівень банківського менеджменту, імідж банку.

Якість активів виражається в низькій ліквідності, яка не дозволяє своєчасно забезпечити приплив грошових коштів. Якість пасивів обумовлюють можливість непередбаченого, дострокового відтоку вкладів і депозитів, що збільшує обсяг вимог до банку в кожен даний момент. Незбалансованість активів та пасивів за строками, сумами і в розрізі окремих валют не у всіх випадках становить загрозу ліквідності. Якщо рівень цієї незбалансованості не виходить за критичні точки, і якщо має місце різнохарактерна спрямованість відхилень в наступні періоди, ризик ліквідності мінімальний.

Процентний ризик відноситься до тих видів ризику, яких банк не може уникнути у своїй діяльності. Більш того, відповідальність за вимір, аналіз і управління ним повністю лежить на менеджменті кредитної організації. Органи нагляду обмежуються, в основному, оцінкою ефективності створеної в комерційному банку системи управління ризиками. Фактори процентного ризику можна поділити на внутрішні і зовнішні. У російській економіці на відміну від розвинених країн рівень ризику підсилюють в основному зовнішні фактори.

До них відносяться:

- нестабільність ринкової кон'юнктури в частині відсоткового ризику;
- правове регулювання процентного ризику;
- політичні умови;
- економічна обстановка в країні;
- конкуренція на ринку банківських послуг;
- взаємини з партнерами і клієнтами;
- міжнародні події.

До внутрішніх факторів процентного ризику можна віднести:

- відсутність чіткої стратегії банку в області центральної виконавчої влади;

- прорахунки в управлінні банківськими операціями, що призводять до створення ризикових позицій (виникнення незбалансованості структури і термінів погашення активів і пасивів, невірні прогнози зміни кривої дохідності тощо);
- відсутність розробленої програми хеджування процентних ризиків;
- недоліки планування і прогнозування розвитку банку;
- помилки персоналу при здійсненні операцій.

Сутність кредитного портфеля банку можна розглядати на категоріальному і прикладному рівнях. У першому аспекті кредитний портфель - це відносини між банком і його контрагентами з приводу поворотного руху вартості, які мають форму вимог кредитного характеру. У другому аспекті кредитний портфель являє собою сукупність активів банку у вигляді позик, врахованих векселів, міжбанківських кредитів, депозитів та інших вимог кредитного характеру, класифікованих за групами якості на основі певних критеріїв.

Якісна відмінність кредитного портфеля від інших портфелів комерційного банку полягає в таких сутнісних властивостях кредиту і категорій кредитного характеру, як зворотний рух вартості між учасниками відносин, а також грошовий характер об'єкта відносин. [2]

1.2 Поняття якості кредитного портфеля

Найважливішим показником рівня організації кредитного процесу є якість кредитного портфеля. Для розкриття змісту якості кредитного портфеля звернемося до тлумачення терміну «якість». Якість - це:

- властивість або приналежність, все, що складає сутність особи або речі;
- сукупність істотних ознак, властивостей, особливостей, що відрізняють предмет або явище від інших і додають йому визначеність;

- та чи інша властивість, ознака, що визначає гідність чого-небудь.

Отже, якість явища має показувати його відміну від інших явищ і визначати його гідність.

Сукупність видів операцій і використовуваних інструментів грошового ринку, що утворює кредитний портфель, має риси, які визначаються характером і метою діяльності банку на фінансовому ринку. Відомо, що позичкові операції та інші операції кредитного характеру відрізняються високим ризиком. У той же час вони повинні відповідати меті діяльності банку - отримання максимального прибутку при допустимому рівні ліквідності. З цього випливають такі властивості кредитного портфеля, як кредитний ризик, прибутковість і ліквідность. Під якістю кредитного портфеля можна розуміти таку властивість його структури, яке має здатність забезпечувати максимальний рівень прибутковості при допустимому рівні кредитного ризику і ліквідності балансу.

Розглянемо зміст окремих критеріїв оцінки якості кредитного портфеля.

Ступінь кредитного ризику. Кредитний ризик, пов'язаний з кредитним портфелем, - це ризик втрат, які виникають внаслідок дефолту у кредитора або контрагента, що носить сукупний характер. Кредитний портфель, як уже зазначалося, має сегменти: позички, надані юридичним, фізичним, фінансовим організаціям; факторингова заборгованість; видані гарантії, враховані векселі та ін.

Оцінка ступеня ризику кредитного портфеля має такі особливості. По-перше, сукупний ризик залежить:

- від ступеня кредитного ризику окремих сегментів портфеля, методики оцінки якого мають як загальні риси, так і особливості, пов'язані зі специфікою сегмента;

- диверсифікованості структури кредитного портфеля і окремих його сегментів.

По-друге, для оцінки ступеня кредитного ризику повинна застосовуватися система показників, що враховує безліч аспектів, які слід взяти до уваги.

Рівень прибутковості кредитного портфеля. Оскільки метою функціонування банку є отримання максимального прибутку при допустимому рівні ризиків, прибутковість кредитного портфеля є одним з критеріїв оцінки його якості. Елементи кредитного портфеля можна розділити на дві групи: що приносять і не приносять дохід активи. До останньої групи належать безвідсоткові кредити, позики з замороженими відсотками і з тривалою простроченням по процентних платежах. У зарубіжній практиці при тривалому простроченій борг по відсоткам практикується відмова від їх нарахування, так як головним є повернення основного боргу. У російській практиці регламентується обов'язкове нарахування відсотків. Рівень прибутковості кредитного портфеля визначається не тільки рівнем процентної ставки за наданими кредитами, а й своєчасністю сплати відсотків і суми основного боргу. [1]

Прибутковість кредитного портфеля має нижню і верхню межу. Нижня межа визначається собівартістю здійснення кредитних операцій (витрати на персонал, ведення позичкових рахунків і т.д.) плюс відсоток, який підлягає сплаті за ресурси, вкладені в цей портфель. Верхньою межею є рівень достатньої маржі.

Рівень ліквідності кредитного портфеля. Оскільки рівень ліквідності банку визначається якістю його активів і, перш за все, якістю кредитного портфеля, то дуже важливо, щоб надані банком кредити поверталися в встановлені договорами терміни або банк мав би можливість продати позики або їх частину, завдяки їх якості та прибутковості. Чим вища частка кредитів, класифікованих в кращі групи, тим вище ліквідність банку.

На користь застосування запропонованих критеріїв оцінки якості кредитного портфеля (ступінь кредитного ризику, рівень прибутковості і

ліквідності) можна навести такі аргументи. Низький ризик елементів кредитного портфеля не означає його високу якість: позики першої категорії якості, які надаються першокласним позичальникам під невеликі відсотки, не можуть приносити високого доходу. Висока ліквідність, притаманна короткотерміновим активам кредитного характеру, також приносить невисокий відсотковий дохід.

Таким чином, кредитний ризик не може бути єдиним критерієм якості кредитного портфеля, оскільки поняття якості кредитного портфеля значно ширше і пов'язане з ризиками ліквідності і втрати прибутковості. Однак значимість названих критеріїв буде змінюватися від умов, місця функціонування банку, його стратегії. [1]

1.3 Управління якістю кредитного портфеля

В управлінні кредитним портфелем велике значення має зміна системи управління термінами активів і пасивів і, отже, різницею процентних ставок і в кінцевому рахунку, прибутковістю. Кожен джерело ресурсів має свої унікальні характеристики, мінливістю і резервними вимогами. Підхід до їх управління - метод конверсії фінансових ресурсів, який розглядає кожне джерело коштів індивідуально.

Управління кредитним портфелем має кілька етапів:

- визначення основних класифікаційних груп кредитів і ставить їх коефіцієнтів ризику;
- віднесення кожного виданого кредиту до однієї із зазначених груп;
- з'ясування структури портфеля (часток різних груп в їх загальній сумі);
- оцінка якості портфеля в цілому;
- виявлення та аналіз факторів, що змінюють структуру (якість) портфеля;

- визначення величини резервів, які необхідно створити під кожен виданий кредит (крім кредитів, під які може бути створений єдиний резерв);
- визначення загальної суми резервів, адекватної сукупного ризику портфеля;
- розробка заходів, спрямованих на поліпшення якості портфеля.

Ключовим моментом в управлінні кредитним портфелем банку є вибір критерію (критеріїв) оцінки якості кожного кредиту і всієї їх сукупності.

Формування резерву обумовлено кредитними ризиками в діяльності банків. Банк формує резерв під можливе знецінення позики (кредиту), тобто під можливу втрату позичкою вартості (повністю або частково) внаслідок реалізовавшегося пов'язаного з даною позикою кредитного ризику. Величина такого знецінення визначається як різниця між балансовою оцінкою позики (залишок заборгованості за позикою, відбитий на рахунках бухгалтерського обліку банку на момент її оцінки) і її так званої справедливої вартістю на момент оцінки (поточна ринкова оцінка позики). При цьому справедлива вартість позики повинна оцінюватися на постійній основі починаючи з моменту видачі позики.

Формуючи резерв, банк, виходячи з категорії позики, визначає розмір так званого розрахункового резерву, тобто резерву, що відображає величину його можливих фінансових втрат по позиці, які будуть визнані такими при дотриманні передбаченого в Положенні порядку оцінки факторів кредитного ризику, але без обліку наявності та якості забезпечення позики.

З метою визначення розміру розрахункового резерву у зв'язку з очікуваним дією факторів кредитного ризику позички (за винятком позик, згрупованих в однорідні портфелі) класифікуються в одну з 5 категорій якості:

- I (вища) категорія якості (стандартні позики) - немає кредитного ризику (ймовірність знецінення позики дорівнює нулю);
- II категорія якості (нестандартні позики) - є помірний кредитний ризик (є ймовірність знецінення позики на 1-20%);

- III категорія якості (сумнівні позики) - є значний кредитний ризик (є ймовірність знецінення позики на 21-50%);
- IV категорія якості (проблемні позики) - присутній високий кредитний ризик (є ймовірність знецінення позики на 51-100%);
- V (нижча) категорія якості (безнадійні позики) - відсутня ймовірність повернення позики, тобто вона буде знецінена повністю (на 100%).

В залежності від якості обслуговування позичальником боргу позики слід відносити в одну з трьох категорій: добре обслуговується; середньо обслуговується; незадовільно обслуговується.

Обслуговування боргу по позиці може бути визнано добрим, якщо:

- платежі за основним боргом і відсотками здійснюються своєчасно і в повному обсязі;
- є тільки одиничний випадок прострочених платежів за основним боргом та / або відсоткам протягом останніх 180 календарних днів, в тому числі:
- за позиками, наданими юридичним особам, - до 5 календарних днів;
- за позиками, наданими фізичним особам, - до 30 календарних днів.

Обслуговування боргу має бути визнано незадовільним, якщо:

- є прострочені платежі за основним боргом та / або за відсотками протягом останніх 180 календарних днів:
- за позиками, наданими юридичним особам, - понад 30 днів;
- наданими фізичним особам, - понад 60 днів;
- позика реструктуровано і по ній є прострочені платежі за основним боргом та / або за відсотками, а фінансове становище позичальника оцінюється як погане;
- позика надана позичальникові прямо або побічно (через третіх осіб) з метою погашення боргу за раніше отриманою позикою, або банк прямо або побічно прийняв на себе ризик втрат у зв'язку з наданням грошей позичальникові, чиє фінансове становище не може бути оцінений вище

середнього за умови, що раніше видана позика була віднесена за якістю обслуговування боргу до категорії позик із середнім обслуговуванням, або при наявності прострочених платежів за новою позикою. [2]

Сформульовані професійні судження про фінансове становище позичальника і про якість обслуговування ним боргу дозволяють шляхом комбінацій двох даних критеріїв визначити категорію якості кожної конкретної позики так, як представлено в таблиці 1.1.

Таблиця 1.1

Обслуговування боргу / Фінансове становище	Добре	Середнє	Незадовільне
Добре	I категорія якості Стандартні позики	II категорія якості Нестандартні суди	III категорія якості Сумнівні позики
Середнє	II категорія якості Нестандартні суди	III категорія якості Сумнівні позики	IV категорія якості Проблемні позики
Погане	III категорія якості Сумнівні позики	IV категорія якості Проблемні позики	V категорія якості Безнадійні позики

Висновки до розділу

У даному розділі було розглянуто поняття кредитного портфелю та методи його аналізу. Особлива увага була приділена підходам, які можна ефективно використовувати для оцінки кредитного портфелю.

Наведена інформація, що дає нам змогу вибрати найбільш ефективні та сучасні методи аналізу кредитного портфелю та використати їх при розробці аналітичної системі підтримки прийняття рішень для управління кредитною заборгованістю. Виходячи з аналізу існуючих підходів до вирішення поставленної задачі, зроблено висновок щодо необхідності розробки підходу для управління кредитним портфелем, що базується на алгоритмах машиного навчання з метою підвищити якість аналізу та ефективності управління кредитною заборгованністю.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ АНАЛІЗУ КРЕДИТНОГО ПОРТФЕЛЯ БАНКУ

2.1 Математичні моделі аналізу кредитного портфеля банку

Математичні моделі формування портфеля банку належать до так званим приватним моделям банківської діяльності, що описує конкретну сферу діяльності банку. З достатньою мірою умовності банк може бути розглянуто як різновид фірми, що функціонує на ринку грошей. У науковій літературі ця обставина знайшло своє відображення в усталеному терміні: «банківська фірма». У зв'язку з цим при моделюванні діяльності банку поряд з іншими методами правомірно використовувати основні поняття і моделі теорії фірми. Чи не випадковим є значна питома вага в загальній кількості математичних досліджень саме моделей фірми, адаптованих до специфіки банківської справи.

При формуванні кредитного портфеля банку виходять з гіпотези про малу керованості ринку депозитів: банк тільки приймає грошові вклади, загальний потік, яких залежить від економічної ситуації в цілому, добробуту населення і т.д., тобто від тих факторів, які знаходяться поза сферою компетенції банку і тому повинні вважатися заданими екзогенно.

Формально модель може бути записана наступним чином.

Цільова функція:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} l_{ij}x_{ij} - \sum_{k=1}^{q} d_k D_k \to \max$$

При обмеженнях:

$$\sum_{i=1}^{m}\left(A_i + R_i\right) \le SK + \sum_{k=1}^{q} D_k$$

$$r_i A_i \le R_i, \; i = \overline{1,m}$$

$$\sum_{j=1}^{n} x_{ij} \le A_i \quad i = \overline{1,m}$$

$$x_{ij} \ge 0, \; i = \overline{1,m}, \; j = \overline{1,n}.$$

Слід зазначити, що в умовах перехідного періоду актуальним є включення в модель чинника ризику. Так як перехідні процеси зазвичай характеризуються, по-перше, високими темпами зміни інфляції, а по-друге, нестабільністю економічних процесів і відсутністю усталених правових та етичних норм бізнесу, то найбільш важливими видами ризиків є ризики процентної ставки та неповернення кредиту; при цьому при моделюванні застосовується зазвичай імовірнісний підхід.

При формуванні кредитного портфеля, банк отримує дохід, пов'язаний з ризиком неповернення виданих позичок. Це дозволяє розглядати формування кредитного портфеля як створення інвестиційного портфеля. Основною моделлю сучасної теорії створення портфеля є модель Марковіца.

Підхід Марковіца починається з припущення, що інвестор в даний момент часу має конкретну суму грошових коштів для інвестування (придбання активів). Ці гроші будуть інвестовані на певний проміжок часу, який називається періодом володіння. В кінці періоду володіння інвестор продає активи, які були куплені на початку періоду. Таким чином, підхід Марковіца може бути розглянуто як дискретний підхід. У початковий момент часу інвестор повинен прийняти рішення про покупку конкретних активів, які будуть перебувати в його портфелі до кінця періоду володіння. Оскільки портфель являє собою набір різних активів, це рішення еквівалентно вибору оптимального портфеля з набору можливих портфелів.

Приймаючи рішення на початку періоду, інвестор повинен мати на увазі, що прибутковість активів, а, значить, і прибутковість портфеля в майбутній період володіння невідома. Однак інвестор може оцінити очікувану (або

середню) дохідність різних активів, грунтуючись на деяких припущеннях. Марковіц зазначає, що типовий інвестор хоча і бажає, щоб «прибутковість була високою», але одночасно хоче, щоб «прибутковість була б настільки певної, наскільки це можливо». Це означає, що інвестор, прагнучи одночасно максимізувати очікувану прибутковість і мінімізувати невизначеність (ризик), має дві суперечать один одному цілі, які повинні бути збалансовані при ухваленні рішення про покупку на початку періоду.

Корисна міра ризику повинна певним чином враховувати ймовірність можливих «поганих» результатів і їх величину. Замість того, щоб вимірювати ймовірності різних результатів, міра ризику повинна певним чином оцінювати ступінь можливого відхилення дійсного результату від очікуваного. Стандартне відхилення - міра, що дозволяє це зробити, так як вона є оцінкою ймовірного відхилення фактичної дохідності від очікуваного.

У загальному випадку обчислення стандартного відхилення портфеля, що складається з n активів, обчислюється за формулою

$$\sigma_p = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} X_i X_j \sigma_{ij}},$$

де X_i - частка вартості портфеля, інвестована в актив і; σ_{ij} - коваріація доходностей активів і та j ($i, j = \overline{1,n}$).

Очікувану прибутковість портфеля можна визначити з співвідношення:

$$\bar{r}_p = \sum_{i=1}^{n} X_i \bar{r}_i$$

де $\bar{r}_i$ – очікувана прибутковість активу i, $i, j = \overline{1,n}$

При цьому повинна виконуватися умова :

$$\sum_{i=1}^{n} X_i = 1$$

Позначимо R - прибутковість портфеля, бажану інвестором. Тоді формально модель створення портфеля може бути записана в наступному вигляді

$$\sigma_p = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} X_i X_j \sigma_{ij}} \to \min$$

$$\bar{r}_p = \sum_{i=1}^{n} X_i \bar{r}_i \geq R$$

$$\sum_{i=1}^{n} X_i = 1$$

$$X_i \geq 0$$

Таким чином, підхід Марковіца до проблеми вибору портфеля передбачає, що інвестор намагається вирішити дві проблеми: максимізувати очікувану прибутковість при заданому рівні ризику і мінімізувати невизначеність (ризик) при заданому рівні очікуваної прибутковості. При цьому очікувана прибутковість служить мірою потенційного винагороди, що з портфелем, а стандартне відхилення сприймається як міра ризику портфеля.

Однак застосування підходу Марковіца до формування кредитного портфеля банку, не дивлячись на схожість завдань, обмежена. Труднощі застосування викликає складний математичний апарат, а також наявність розвиненої системи збору інформації для реалізації моделі.

2.2 Статистичні методи аналізу кредитного портфеля банку

2.2.1 Використання статистичних методів

Одним з основних елементів процесу оцінки майна є збір інформації, зокрема, про їх ринкових цінах. Як правило, тут оцінювач стикається з таким

явищем, коли ціни практично ідентична ого майна , отримані з різних джерел, відрізняються один від одного. У цьому випадку говорять, що зібрані оцінювачем значення цін є випадковими (або стохастичними) величинами. На основі зібраної інформації оцінювачу в цьому випадку доводиться визначати статистичні оцінки ряду величин, зокрема, середнє значення ринкової ціни об'єкта оцінки. При малих обсягах зібраної інформації оцінювач повинен бути впевнений в її якості. Тому вже на стадії попередньої обробки інформації він повинен провести відсів різко виділяються спостережень в вибірці і перевірку гіпотези про нормальність розподілу. Тільки після цього можливе застосування методів і співвідношень, добре розроблених для нормального розподілу.

Наступним кроком є оцінка похибки середнього значення ціни з використанням, наприклад, інтервальних оцінок. Далі при побудові моделі ціни об'єкта оцінювачу необхідно, використовуючи кореляційні методи, оцінити ступінь впливу на неї різних факторів, провести класифікацію факторів і, нарешті, побудувати саму модель у вигляді рівняння регресії. Такий далеко не повний перелік завдань, коли оцінювачу може знадобитися математична статистика.

2.2.2 Основні статистичні характеристики

Отже, інформація, з якою доводиться працювати оцінювачу, в значній мірі відноситься до категорії випадкових величин.

Випадковою величиною називають таку величину, значення якої змінюються деяким, заздалегідь не передбачуваним чином. На відміну від невипадкових, детермінованих величин для випадкової величини можна заздалегідь точно сказати, яке конкретне значення вона прийме в певних умовах, а можна тільки вказати закон її розподілу. Законом розподілу називають сукупність значень випадкової величини і ймовірностей, з якими вона їх приймає. Сума всіх ймовірностей завжди дорівнює одиниці, так як з такою ймовірністю величина приймає хоч яке-небудь з цих значень. Існує

багато причин, що призводять до того, що значення ринкових цін у вибірці виявляються швидше випадковими, ніж детермінованими. Часто це викликано відсутністю інформації про всі фактори, що впливають на ціну майна , або нечіткістю цієї інформації. Наприклад, як у випадку нечіткості інформації про ступінь фізичного зносу майна , недостатності даних про умови угоди купівлі-продажу та т. П. Неконтрольовані фактори можуть брати випадкові значення з деякої множини значень і тим самим обумовлювати випадковість тих величин (зокрема, цін), які вони визначають. Тому справжнє значення ціни майна виявляється недоступним оцінювачу, і навіть усереднення випадкових значень цін у вибірці не усуває випадковості середнього значення ціни. Стохастична природа даних, що використовуються оцінювачем в процесі визначення вартостей об'єктів, викликає необхідність застосування адекватних їм статистичних методів аналізу. Базою для застосування статистичних методів аналізу при оцінці зазвичай є безліч емпіричних даних, отриманих за результатами збору інформації про одну або кількох випадкових величинах (цінах близьких аналогів об'єкта оцінки, ступеня їх зносу, витрати на ремонт і т.п.). Будемо позначати їх великими латинськими літерами X , Y , Z ... Інформація про будь-який з цих величин складається з n значень цієї випадкової величини X , що утворюють вибірку обсягу n з генеральної сукупності X.

Під генеральною сукупністю маються на увазі всі можливі значення конкретної випадкової величини (наприклад, ринкової ціни машини).

Зібрати дані про всі значеннях з генеральної сукупності практично неможливо. Тому реально оцінювач задовольняється вибіркою, а методи математичної статистики допомагають йому по відомим властивостям об'єктів з вибірки судити про властивості всієї генеральної сукупності. При використанні даних вибірки через випадкового характеру її отримання важливо знати, яким імовірнісним законам підкоряються значення досліджуваного показника. Існує цілий ряд розподілів ймовірності, які використовуються в

математичній статистиці. Одним з найбільш часто використовуваних розподілів і тому важливих є нормальний розподіл. Теоретичним обґрунтуванням ролі нормального розподілу є центральна гранична теорема. Відповідно до цієї теореми, розподіл середнього n незалежних випадкових величин, розподілених по будь-якому закону, при збільшенні числа значень у вибірці наближається до нормального. Коли випадкова величина являє собою загальний результат великого числа незалежних «невеликих» впливів (маються на увазі вплив неконтрольованих факторів), то, згідно з центральною граничною теоремою, можна очікувати, що ця випадкова величина буде розподілена за нормальним законом.

Випадкова величина X має нормальний розподіл, якщо її щільність ймовірності описується рівнянням (при $-\infty < x < \infty$)

При описі випадкової величини замість закону розподілу можна використовувати його параметри μ і σ2 - відповідно математичне сподівання випадкової величини і її дисперсію. Якщо відомі параметри розподілу, то щільність ймовірності повністю визначена.

Однак на практиці оцінювач завжди користується даними вибірки з генеральної сукупності даних. У цьому випадку деякі основні властивості випадкових величин можуть бути описані більш просто за даними вибірки за допомогою оцінок параметрів їх функцій розподілу, званих також статистиками. Найважливішими з цих оцінок є: середнє (середнє арифметичне) значення вибірки (оцінка математичного очікування).

Стандартне відхилення s - міра розкиду випадкової величини навколо середнього значення, що має розмірність, збігається з розмірністю випадкової величини, що корисно при визначенні похибок в оцінках. Поряд зі згаданими статистиками для опису сукупності даних використовують і інші.

Медіана, або серединна значення, розділяє випадкові величини на рівні половини. Для її обчислення всі зібрані дані потрібно розташувати в порядку зростання або зменшення. Потім, якщо n - непарне число, то медіану

визначають як значення, що знаходиться в середині упорядкованого послідовності. При парному n медіана - середнє арифметичне двох розташованих в середині значень впорядкованої послідовності. Мода - є найбільш часто зустрічається в сукупності даних величина.

До характеристик розкиду даних відноситься також коефіцієнт варіації вибірки:

$$\nu = \frac{s}{x_{cp}} \cdot 100\%$$

,

Значення ν висловлює середньоквадратичне відхилення s в процентах від середнього сукупності даних і тому може бути використано для оцінки їх точності.

Розглянуті вище характеристики випадкових величин є так званими точковими оцінками відповідних їм характеристик генеральної сукупності.

Статистичні оцінки обчислюють виходячи з конкретного закону розподілу випадкової величини. Зазвичай передбачається, що ціна як випадкова величина підпорядковується закону нормального розподілу. Це, як правило, обгрунтовується в разі оцінки центральною граничною теоремою. Однак процедура формування оцінювачем малої вибірки ринкових цін з генеральної сукупності не може гарантувати її однорідності. Тому на початковій стадії обробки даних бажано проведення перевірки гіпотези нормальності розподілу вибіркових даних про ціни ідентичних об'єктів. Це дозволить оцінювачу більш обгрунтовано застосовувати статистичні оцінки даних, відповідні цим законом. У математичній статистиці існує ряд методів перевірки нормальності розподілу. Найбільш відомим з них є чисельний метод застосування критерію , розроблений К. Пірсоном. Однак малі вибірки, з якими зазвичай має справу оцінювач, не можуть дати достатньої кількості даних для застосування таких критеріїв. Тому покажемо тут більш грубі методи, що дозволяють судити про нормальність розподілу малої вибірки.

У математичній статистиці поряд з точковими оцінками широко використовуються так звані інтервальні оцінки - інтервали між статистиками, що містять з певною ймовірністю істинне значення оцінюваного параметра. Для побудови інтервального оцінки параметра (наприклад, середньої ціни ЦСР) необхідно знайти дві статистики L і U такі, при яких справедливо вірогідне твердження:

$$P\left(L \le Ц_{ср} \le U\right) = 1 - \alpha$$

Інтервал $L \le Ц_{ср} \le U$ називається процентним довірчим інтервалом. Цьому інтервалу можна дати наступну інтерпретацію: з ймовірністю (1 - α) в зазначеному інтервалі буде знаходитися справжнє значення ціни. Статистики L і U називаються нижньої і верхньої довірчими межами інтервалу відповідно, величина (1 - α) - довірчою ймовірністю, а величина α - рівнем значущості (ймовірністю помилки). Якщо α = 0,1, то інтервал називається 90-процентним довірчим інтервалом.

2.2.3 Показники якості кредитного портфеля

Оцінка якості кредитного портфеля проводиться за показниками ризику кредитного портфеля та показниками дохідності кредитних операцій та базується на коефіцієнтах наведених у таблиці 2.1.

Таблиця 2.1 - Коефіцієнти оцінки якості кредитного портфеля

Показник	Розрахунок	Значення показника
Коефіцієнт ризику (Р)	Р = (КВ - ПрВ) / КВ, де КВ - сукупність кредитних вкладень банку, ПрВ - прогнозовані втрати банку (прогнозовані втрати банку на звітну дату визначаються як сукупна величина резервів на можливі	Коефіцієнт дозволяє найбільш чітко визначити якість кредитного портфеля з позиції кредитного ризику, проте інтерпретація його двояка. Чим ближче значення коефіцієнта ризику кредитного портфеля до 1, тим краще якість кредитного портфеля з точки зору повернення (відновлення) виданих позичок; це також дозволяє говорити про те, що кредитний

	втрати з позик, позикової і прирівняної до неї заборгованості	портфель сформований за рахунок кредитів «підвищеної якості» (стандартних і нестандартних) [5]

Продовження таблиці 2.1

Коефіцієнт проблемності кредитів (Кпр)	Кпр = ПК/КВ, де ПК - сума простроченого основного боргу, КВ - загальний обсяг кредитного портфеля	Roефіцієнт показує, яка частка прострочених платежів за основним боргу припадає на одну гривню кредитного портфеля [5]
Коефіцієнт покриття збитків за позиками (Кпз)	Кпз = РМВП / Квпр, де РМВП - резерв на можливі втрати по позиках, Квпр - величина простроченої позичкової заборгованості	Дозволяє визначити рівень покриття проблемних кредитів. Рекомендоване значення Кпз> 1
Коефіцієнт покриття кредитного портфеля власним капіталом банку (Квк)	Квк = ВК/КВ, де ВК - власний капітал, КВ - сукупність кредитних вкладень банку	Характеризує якість кредитного портфеля з погляду його захищеності власним капіталом і показує, скільки власних коштів припадає на одну гривню виданих кредитів [5]
Коефіцієнт покриття класифікованих кредитів власним капіталом (Кпвк)	Кпвк = РМВП/ВК, де РМВП - резерв на можливі втрати по позиках, КВ - сукупність кредитних вкладень банку	Коефіцієнт характеризує якість кредитного портфеля банку з погляду його захищеності власним капіталом банку [5]
Коефіцієнт дохідності кредитного портфеля (Кд)	Кд = ВД/КВ, де ВД – відсотковий дохід банку, КВ - сукупність кредитних вкладень банку	Коефіцієнт характеризує доходність вкладень у кредитний портфель і показує, скільки банк отримує доходів з кожної гривні виданих кредитів [5]
Коефіцієнт прибутковості кредитних операцій (Кпко)	Кпко = ЧВД/КВ, де ЧВД – чистий відсотковий дохід банку, КВ - сукупність кредитних	Коефіцієнт показує, скільки відсоткового прибутку припадає на одну гривню розміщених у кредитний портфель ресурсів [5]

	вкладень банку	

Продовження таблиці 2.1

Коефіцієнт забезпечення (Кз)	розраховується як відношення суми забезпечення, прийнятої банком при видачі кредиту, до загальної суми кредитного портфеля	Rоефіцієнт показує, яка частка забезпечення повернення кредитів припадає на одну гривню кредитного портфеля. Рекомендоване значення показника: Кз > 1 [5]

Свої особливості мають процедури оцінки кредитного ризику і визначення суми резерву по позиках, згрупованих в однорідний портфель. Реально резерв формується з урахуванням наявності і категорії забезпечення позики. Розрахунковий резерв отримується шляхом підсумування залишку заборгованності по кредиту помноженої на коефіцієнт класу. Для наших классів маємо такі коефіцієнти:

Таблиця 2.2 - Величина розрахункового резерву за класифікованими позиками

Категорія якості	Найменування	Розмір розрахункового резерву,% від суми основного боргу за позикою
I категорія якості	стандартні	1
II категорія якості	нестандартні	10
III категорія якості	сумнівні	30
IV категорія якості	проблемні	60
V категорія якості	безнадійні	100

2.3 Оцінка динаміки та ефективності кредитних портфелів банків України

У сучасних умовах стан банківської системи України є складним і важко прогнозованим, що обумовлено як трансформаційними процесами в економіці країни, так і складними політичними та соціальними умовами. Проте, однією з важливих складових розвитку економіки держави є стабільність та стійкість банківської системи. Можливість прогнозувати стан банківської системи дає змогу збільшити привабливість її для іноземних інвесторів та вітчизняних суб'єктів господарювання, а також визначити основні напрямки подальшого розвитку [12]. Ефективність кредитної діяльності банків на сьогодні є необхідним, якщо не вирішальним, фактором життєдіяльності банків, оскільки кредитний портфель становить більше половини всіх активів банку. У структурі балансу банку кредитний портфель розглядається як єдине ціле та складова частина активів, що має свій рівень дохідності та ризику. Тому для успішного кредитування банки мають сформувати та реалізувати ефективну систему управління кредитним портфелем. Сучасні умови підтверджують, що обмежені можливості залучення кредитних коштів також впливають на розвиток економіки держави. Активізація кредитування сприяє подальшому зростанню економіки, появі нових суб'єктів господарювання, збільшенню робочих місць, розвитку інфраструктури країни та ін., що створює основу для економічної стабільності. При цьому, кредитування є одним з магістральних напрямків діяльності банків, який забезпечує їм прибутковість. Але процес кредитування для банків супроводжується певними ризиками, одним з яких є ризик неповернення запозичених коштів [13]. Для розвитку процесу кредитування варто спочатку проаналізувати сучасний стан кредитних операцій банку, починаючи з визначення місця, яке посідають кредитні операції в

загальному обсязі активів банку. Для цього розраховується коефіцієнт питомої ваги кредитів у загальних активах банку за формулою (1) [14]:

$$Ч_{К.О.} = \frac{\text{Середні кредитні вкладення}}{\text{Середні активи}} (1)$$

Для розрахунку даного коефіцієнту використаємо показники кредитних вкладень банків України за період 2013 – 2017 рр., кількості банків та їхніх активів. Вхідні дані наведено у таблиці 2.3.

Таблиця 2.3 - Вхідні дані для розрахунку коефіцієнта питомої ваги кредитів у загальних активах банку (на основі статистичних даних [11])

Дата	Кількість банків, шт	Активи у нац. валюті, тис. грн	Кредити та заборгованість клієнтів, тис. грн
01.01.2013	175	1127179379	694 381 045
01.01.2014	180	1277508651	799 227 946
01.01.2015	158	1316717870	873 610 836
01.01.2016	113	1252570443	713 974 266
01.01.2017	95	1258643603	554 637 262

Підставивши данні таблиці 2.3 у формулу (1), отримаємо значення коефіцієнта питомої ваги кредитів у загальних активах окремого банку, яке становить 59,88%. Таке значення свідчить про те, що частка кредитних операцій у загальних активах банку складає 59,88% і характеризує кредитну активність банків. Достатньо велике значення цього показника може вказувати на те, що банки занадто перевантажені позиками. Існує думка, що значення цього показника понад 65% є зависоким. Проте ліквідні банки мають нижчий рівень цього коефіцієнта, але більшу частку ресурсів у короткострокових коштах грошового ринку та в інвестиційних цінних паперах, які, у свою чергу, можуть легко конвертуватися в грошові кошти та потім видаватися у формі позик [14]. Оскільки кредитування посідає чільне місце у операціях банків, варто розглянути динаміку кредитних портфелів та їх структуру. На рисунку

2.1 наведено динаміку обсягів кредитних портфелів банків України за 2013-2017 роки.

Рисунок 2.1 – Кредитний портфель та активи банків України 2013-2017 рр. (на основі статистичних даних [11])

З наведених даних можна зробити висновок, що з 2013 р. по 2015 р. кредитний портфель банків України збільшувався, а з 2015 р. по 2017 р. прослідковується його зменшення. Це пов'язано із зменшенням кількості банків в Україні. Проте, активи банків у 2017 році порівняно з 2016 роком зросли і це говорить про те, що на фінансовому ринку залишаються хороші банки, які примножують свої активи та працюють відповідно до законодавства, яке, нажаль, часто змінюється та диктує жорсткі умови функціонування. Для кращого розуміння динаміки кредитного портфеля банків України розрахуємо такі показники: абсолютне відхилення, темп зростання та темп приросту кредитного портфеля банків за формулами (2),(3) та (4).

$$\Delta = \text{Зн.пок.пот.п.} - \text{Зн.пок.поп.п.} \quad (2),$$

де Δ – абсолютне відхилення, Зн.пок.пот.п. – Значення показника у поточному періоді, Зн.пок.поп.п. – Значення показника у попередньому періоді;

$$\text{Тзр} = \frac{\text{Зн.пок.пот.п.}}{\text{Зн.пок.поп.п.}} * 100\% \quad (3),$$

де Тзр – темп зростання;

$$Тпр = \frac{Зн.пок.пот.п.}{Зн.пок.поп.п.} * 100\% - 100 \ (4),$$

де Тпр – темп приросту.

Отримані нами результати наведені у таблиці 2.4.

Таблиця 2.4 - Динаміка деяких показників кредитного портфеля банків України у 2013- 2017рр. (на основі статистичних даних [11])

Дата	Кредити та заборгованість клієнтів, тис. грн	Абсолютне відхилення, тис. грн	Темп зрост.	Темп приросту
01.01.2013	694 381 045	-	-	-
01.01.2014	799 227 946	104846901	115,10%	15,10%
01.01.2015	873 610 836	74382890	109,31%	9,31%
01.01.2016	713 974 266	-159636570	81,73%	-18,27%
01.01.2017	554 637 262	-159337004	77,68%	-22,32%

Як бачимо з таблиці 2.4 у 2014 р. порівняно з 2013 р. відбулося збільшення кредитного портфеля банків України до рівня 115,10% або на 15,10%. У 2015 р. порівняно з 2014 р. також відбулося збільшення – до рівня 109,31% або на 9,31%. У 2016 р. порівняно з 2015 р. значення кредитного портфеля банків України становило 81,73%, тобто відбулося зменшення на 18,27% або на 159 636 570 тис. грн. У 2017 р. порівняно з 2016 р. значення кредитного портфеля банків України становило 77,68%, також відбулося зменшення на 22,32% або на 159337004 тис. грн. У цілому, за період 2013-2017 рр. відбулося зменшення кредитного портфеля банків України на 20,13%. Це наслідки кризових процесів у вітчизняній економіці країни та світу та зменшенням кількості банків на 45,71% (80од.).

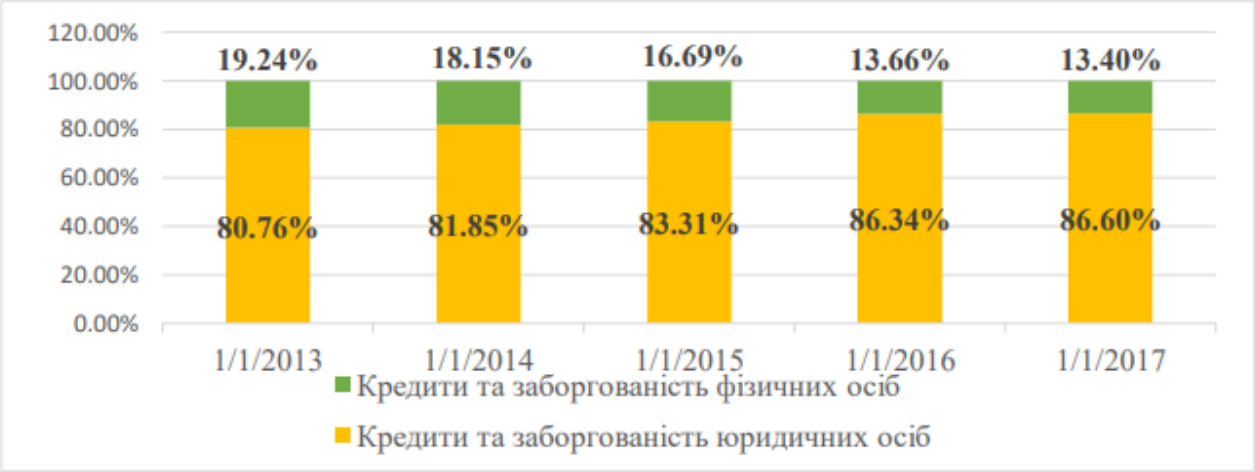


Рисунок 2.2 - Розподіл кредитного портфеля банків України за юридичними та фізичними особами (на основі статистичних даних [11])

Наведена інформація на рисунку 2.2 показує, що більша частина кредитів та заборгованості клієнтів припадає на юридичних осіб. У відсотковому співвідношенні у 2017 р. порівняно з 2013 р. частка кредитів та заборгованості юридичних осіб зросла на 5,84%. Розглянемо також структуру кредитного портфеля у розрізі груп ризику, рівня забезпеченості галузевої структури, форм власності позичальників і т. п., а також проаналізуємо динаміку кожної групи, сегментацію кредитного портфеля[15]. Кредитний портфель банку схильний до всіх основних видів ризику, які супроводжують фінансову діяльність: ризику ліквідності, ризику процентних ставок, кредитному ризику. Банки визначають кредитний ризик за кредитами, наданими юридичним та фізичним особам, визначивши імовірність настання дефолту клієнта. Імовірність дефолту клієнта визначають за фінансовим станом клієнтів, за яким поділяють фізичних осіб на 5 класів (1 – фінансовий стан високий, 2 – фінансовий стан добрий, 3 – фінансовий стан задовільний, 4 – фінансовий стан незадовільний, 5 - фінансовий стан критичний), а юридичних – на 10 класів(залежно від отриманого значення інтегрального показника фінансового стану боржника з урахуванням величини підприємства)[16].

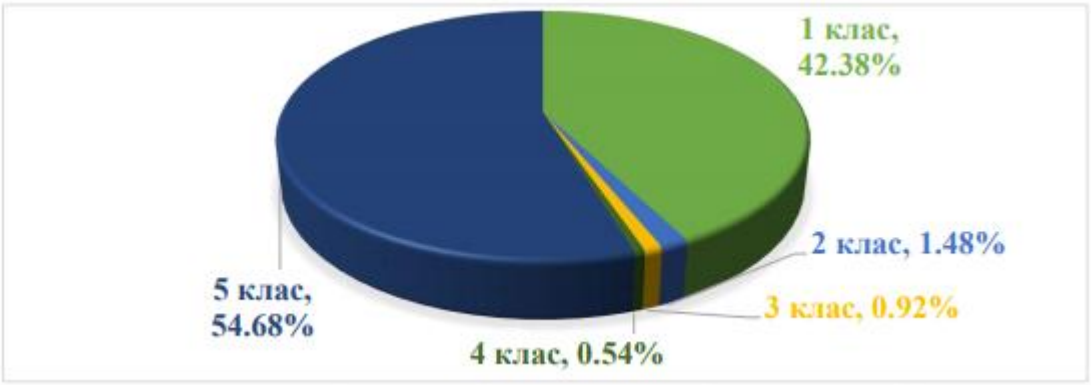


Рисунок 2.3 - Розподіл кредитів за класами боржника – фізичної особи (на основі статистичних даних [11])

На рисунку 2.3 представлено розподіл кредитів за класами боржника – фізичної особи станом на 01.11.2017р., який показує, що найбільшу частку – 54,68% кредитів було видано 5 класу боржника, який характеризується критичним фінансовим станом і несе великий ризик для банків. Проте значну частку – 42,38% становлять кредити, що надані 1 класу боржника, до якого входять надійні позичальники з високим фінансовим станом.

Рисунок 2.4 - Розподіл кредитів за класами боржника – юридичної особи (на основі статистичних даних [11])

На рисунку 2.4 представлено розподіл кредитів за класами боржника – юридичної особи станом на 01.11.2017 р., який показує, що найбільша частка кредитів – 56,91% належить до 10 класу боржника, що свідчить про велику ризикованіість даних кредитів. Натомість 1 класу, який є найменш ризиковим,

було видано лише 1,04%. З наведених даних випливає, що найбільше надано кредитів як фізичним так і юридичним особам останнього класу боржників, які мають велике значення кредитного ризику. Можемо зробити припущення, що така ситуація склалась за рахунок того, що у минулі роки позичальники, які отримали кредити у іноземній валюті, через значні коливання курсу автоматично стали боржниками з критичним фінансовим станом. Крім того, це може свідчити про те, що значну частку кредитів банки видають інсайдерам, а, відповідно до законодавства, такі позичальники відносяться до боржників з високим рівнем ризику.

2.4 Машине навчання у задачі аналізу кредитного портфеля

В аналізі кредитного портфеля описова і діагностична аналітика зарекомендували себе добре, але все-таки ці методи не позбавлені недоліків. Використання аналітики обмежується регуляторами - не всі просунуті методи і моделі можуть бути схвалені з їх боку. Аналітика не відрізняється гнучкістю і не дає можливості представити дані в довільному зрізі - а це часто буває дуже потрібно. Та й з оперативністю в даному випадку не все добре. А ще трапляється, що для роботи якихось аналітичних моделей просто не вистачає даних. Тому є доцільним використовувати машине навчання для розрахунку факторів кредитного ризику.

Машинне навчання — це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися» (тобто, поступово покращувати продуктивність у певній задачі) з даних, без того, щоби бути програмованими явно. [6]

Задачі машинного навчання, як правило, поділяють на дві широкі категорії, залежно від того, чи доступний системі, що навчається, навчальний «сигнал», або «зворотний зв'язок»:

- навчання з учителем: комп'ютерові представляють приклади входів та їхніх бажаних виходів, задані «вчителем», і метою є навчання загального правила, яке відображає входи на виходи. В окремих випадках вхідний сигнал може бути доступним лише частково, або бути обмеженим особливим зворотним зв'язком;
- навчання без учителя: алгоритмові навчання не дається міток, залишаючи його самому знаходити структуру в своєму вході. Навчання без учителя може бути метою саме по собі (виявлення прихованих закономірностей у даних), або засобом досягнення мети (навчання ознак).

Оскільки для аналізу кредитного портфеля необхідно використати методи класифікації, то надалі розглядатимемо лише методи навчання з учителем. Найбільш поширеними методами машинного навчання для задач класифікації є:

- навчання дерев рішень. Навчання дерев рішень використовує як передбачувальну модель дерево рішень, яке відображує спостереження про предмет на висновки про цільове значення предмету;
- штучні нейронні мережі. Алгоритм навчання штучної нейронної мережі, зазвичай званої «нейронною мережею» (НМ), є алгоритмом навчання, віддалено натхненим біологічними нейронними мережами. Обчислення структуруються в термінах взаємозв'язаних груп штучних нейронів, які обробляють інформацію із застосуванням конективістського підходу до обчислень;
- метод опорних векторів. Метод опорних векторів є набором пов'язаних методів навчання з учителем, які використовуються для класифікації та регресії. Маючи набір тренувальних прикладів, кожен з яких помічено як належний до однієї з двох категорій, алгоритм тренування методу опорних векторів будує модель, яка передбачує, чи новий приклад потрапляє до однієї категорії, чи до іншої.

2.4.1 Штучна нейрона мережа

Для вирішення задачі класифікації позик було вирішено використовувати штучну нейрону мережу.

Штучні нейронні мережі (ШНМ) — математичні моделі спроектовані за прикладом дії біологічних нейронних мереж. Основою ШНМ є модель людського мозку, який складається із мільярдів нейронів що з’єднанні синапсами. Аналогічно, штучні нейронні мережі складаються з обчислювальних
елементів які називають штучними нейронами. Біологічний нейрон імітується у ШНМ через активаційну функцію. У задачах класифікації (наприклад визначення спам-повідомлень) активаційна функція повинна мати характеристику "вмикача". Іншими словами, якщо вхід більше, ніж деяке значення, то вихід повинен змінювати стан, наприклад з 0 на 1 або -1 на 1. Це імітує "включення" біологічного нейрону. У якості активаційної функції зазвичай використовують сигмоїдну функцію:

$$A = \frac{1}{1+e^{-x}}$$

Яка виглядає наступним чином:

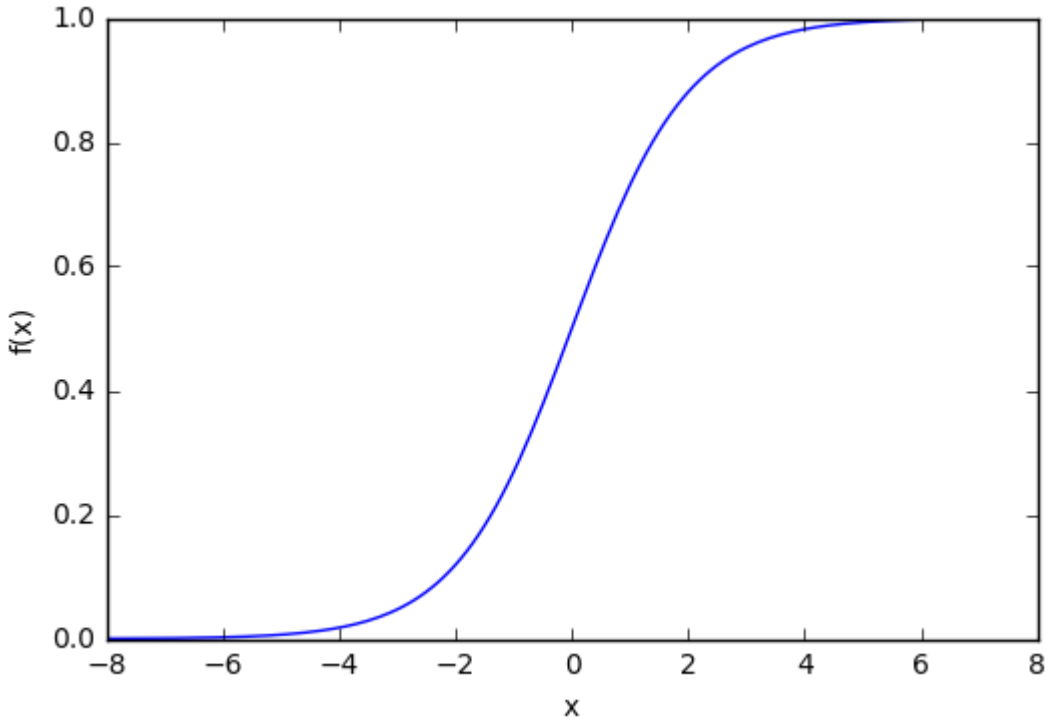


Рисунок 2.5 – Графік сигмоїда

З рисунку 2.5 можна побачити, що функція "активаційна" – вона росте з 0 до 1 з кожним збільшенням значення x. Сигмоїдна функція є гладкою і неперервною. Це означає, що функція має похідну, що у свою чергу є дуже важливим фактором для навчання алгоритму.

Біологічні нейрони ієрархічно з'єднані в мережами, де вихід одних нейронів є входом для інших нейронів. Ми можемо представити такі мережі у вигляді з'єднаних шарів з вузлами. Кожен вузол приймає зважений вхід, активує активаційну функцію для суми входів, та генерує вихід.

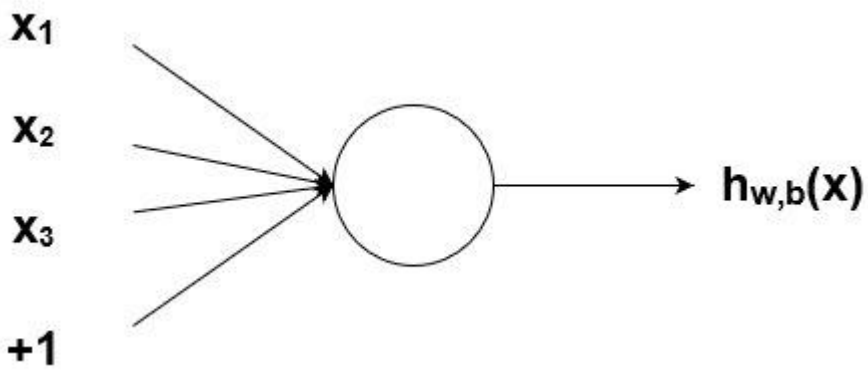


Рисунок 2.6 – Вузол зі входами

На рисунку 2.6 зображено вузол. Вузол є "місцерозташуванням" активаційної функції, він приймає зважені входи, сумує їх, а потім вводить їх в активаційну функцію. Вивід активаційної функції представлений через h. За вагу беруться числа (не бінарні), які потім множаться на вході і сумуються вузлі.

Структури нейроних мереж можуть приймати багато різних форм, але найпоширеніша складається з вхідного шару, прихованого шару та вихідного шару. Приклад такої структури приведено нижче:

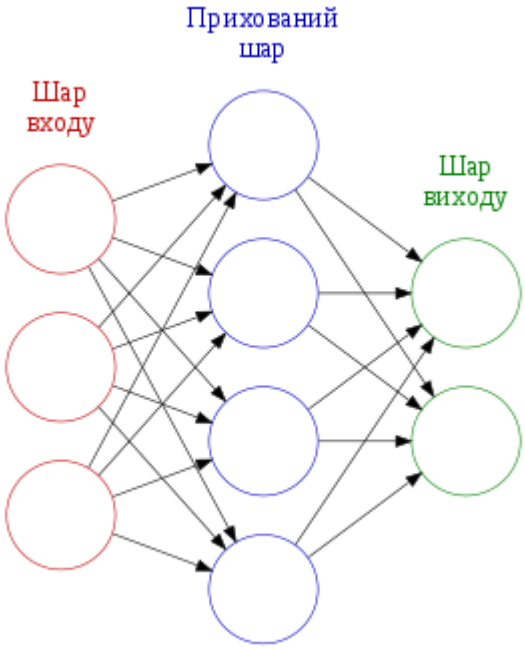


Рисунок 2.7 – Нейрона мереже з одним прихованим шаром

На рисунку 2.7 можна побачити три шари мережі - вхідний шар, де мережа приймає зовнішні вхідні дані, прихований шар, що не є частиною ні входу, ні виходу, та вихідний шар.

2.4.2 Модель нейроної мережі

Спочатку оберемо модель даних, які будуть подаватися на вхід нейроної мережі і даних, які мережа буде генерувати на виході.

Вхідними даними для нашої системи будуть слугувати параметри кредита та інформація клієнта.

Нижче представлена модель даних у тому вигляді, в якому вона буде використана системою.

Таблиця 2.5 – Вхідні параметри

Найменування показника	Одиниця вимірювання
Сума кредиту	Число (грн)
Поточна заборгованність	Число (грн)
Прострочення заборгованність	Число (грн)
Кредитний рейтинг контрагента	Число від 0 до 1000
Кількість днів прострочки	Число
Кількість місяців з останьої прострочки	Число
Кількість відкритих договорів	Число
Кількість кредитних проблем у минулому	Число
Сума платежів по кредиту	Число (грн)
Оцінка залогу	Число (грн)
Кредитний рейтинг бенефіціара	Число від 0 до 1000

Крім цього, користувач може обрати власні вхідні параметри та провести на них навчання нейронної мережі.

Вихідними даними будуть класи кредитів:

- стандартні кредити;
- нестандартні кредити;
- сумнівні кредити;
- проблемні кредити;
- безнадійні кредити.

2.4.3 Методи навчання нейронної мережі

Для роботи нейронної мережі необхідно провести її навчання. Існує три основні парадигми навчання, кожна з яких відповідає певній навчальній задачі. Ними є кероване навчання, спонтанне навчання та навчання з підкріпленням.

Кероване навчання.

Визначення методу керованого навчання можна сформулювати так: задача машинного навчання у реалізації функції виведення на підставі відомих навчальних даних.

Структура навчальних даних складається з набору прикладів для виконання навчання. В даному методі навчання кожен приклад представляє собою пару, що складається з вхідного об'єкта (зазвичай вектора) і бажаного вихідного значення (контрольний сигнал). Необхідно прорахувати значення функції на момент входу даних, потім порівняти значені із значенням очікуваного резузльту, вирахувати помилку і скорегувати параметри (вага нейронів) мережі на цю помилку.

Спонтанне навчання.

Метод спонтаного навчання, також називають методом неконтрольованого навчання, який має змогу знайти структуру або відносини між різними входами. Головна відмінність від методу навчання "з вчителем", це те що в наявності є тільки вхідні дані. Алгоритм навчання без вчителя застосовується по суті тоді, коли відомі тільки вхідні дані. На основі їх мережа вчиться видавати найкращі вихідні результати. Поняттям «найкращих

результатів» визначається самим алгоритмом навчання. Зазвичай алгоритм підлаштовує параметри так, щоб мережа видавала однакові результати для достатньо близьких вхідних значень [8]. Найбільш важливим неконтрольованим навчанням є кластеризація, яка створює різні кластери введення і зможе вносити будь-які нові дані до відповідного кластеру. Крім кластеризації є інші методи неконтрольованого навчання: виявлення аномалій, навчання і навчання в теорії Хебба. Приховані змінні моделі, такі як алгоритм максимізації очікувань, метод моментів і методи поділу сліпих сигналів. Хоче даний метод і часто використовується у прикладних задачах, він часто піддається критиці вчених через свою біологічну схильність. Важко уявити, що у мозку присутній механізм порівняння отриманих результатів з бажаними.

Навчання з підкріпленням.

Навчання з підкріпленням є проміжним варіантом двох попередніх парадигм . Замість «вчителя» в схему навчання вводиться блок «критика» , який відслідковує реакцію середовища на вхідний сигнал і опираючись на неї визначає евристичну похибку, яку покладено в процес навчання мережі. Вказані парадигми базуються на відповідних правилах навчання, які визначають основні особливості їх застосування. [7]

2.4.4 Метод зворотнього поширення помилки

Згорткові мережі за своїм типом поширення активаційного сигналу між нейронами є прямими, тому застосування алгоритму зворотного поширення помилки є цілком доречним до таких мереж. Метод зворотного поширення помилки — метод навчання багатошарового перцептрону. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи багатошарового перцептрону та отримання бажаного виходу. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Барц і Охонін запропонували відразу загальний метод

(«принцип подвійності»), який можна застосувати до ширшого класу систем, включаючи системи з запізненням, розподілені системи, тощо [9]. Для можливості застосування методу зворотного поширення помилки функція активації нейронів повинна бути диференційованою.

Алгоритм:

1. Ініціалізувати $\{w_{ij}\}_{i,j}$ маленькими випадковими значеннями, $\{\Delta w_{ij}\}_{i,j} = 0$
2. Повторити NUMBER_OF_STEPS раз:

   Для всіх d від 1 до m:

   2.1.Подати $\{x_i^d\}$ на вхід мережі і підрахувати виходи o_i кожного вузла.

   2.2.Для всіх $k \in Outputs$

   $$\delta_k = -o_k(1 - o_k)(t_k - o_k)\,.$$

   2.3.Для кожного рівня l, починаючи з передостаннього:

   Для кожного вузла j рівня l обчислити

   $$\delta_j = -o_j(1 - o_j) \sum_{k \in Children(j)} \delta_k w_{j,k}\,.$$

   2.4.Для кожного ребра мережі {i, j}

   $$\Delta w_{i,j} = \alpha \Delta w_{i,j} + (1 - \alpha)\eta \delta_j o_i\,.$$
   $$w_{i,j} = w_{i,j} + \Delta w_{i,j}\,.$$

3. Видати значення w_{ij} .

де α - Коефіцієнт інерціальних для згладжування різких стрибків при переміщенні по поверхні цільової функції

Головними недоліками градієнтного спуску або алгоритму алгоритму зворотнього поширення помилки при навчанні мережі є [10]:

- «параліч» мережі. Значення ваг мережі в результаті корекції можуть досягти дуже великих величин. Оскільки помилка, що посилається назад в процесі навчання, пропорційна похідній стискаючій функції, процес навчання може майже зупинитися. Цьому можна запобігти, зменшуючи крок, але процес навчання при цьому буде відбуватися довше;

- розмір кроку. Якщо значення кроку не змінюється і воно досить мале, то метод сходитися занадто повільно. Якщо ж крок занадто великий, то може виникнути параліч мережі. Необхідно змінювати значення кроку: збільшувати до тих пір, 50 поки не припиниться поліпшення оцінки в напрямок антиградієнта і зменшувати, якщо оцінка не поліпшується.

2.4.5 Ітераційні алгоритми оптимізації

Для мінімізації помилки роботи багатошарового перцептрону можуть використовуватися такі (ітераційні) алгоритми:

- алгоритми локальної оптимізації з обчисленням приватних похідних першого порядку: градієнтний алгоритм (метод найшвидшого спуску), методи з одновимірної і двовимірної оптимізацією цільової функції в напрямку антіградієнта, метод сполучених градієнтів, методи, що враховують напрямок антіградієнта на кілька кроків алгоритму;

- алгоритми локальної оптимізації з обчисленням приватних похідних першого і другого порядку: метод Ньютона, методи оптимізації з розрідженими матрицями Гессе, квазіньютоновскіе методи, метод Гаусса-Ньютона, метод Левенберга-Марквардта та ін .;

- стохастичні алгоритми оптимізації: пошук у випадковому напрямку, імітація відпалу, метод Монте-Карло (чисельний метод статистичних випробувань);

- алгоритми глобальної оптимізації (задачі глобальної оптимізації вирішуються за допомогою перебору значень змінних, від яких залежить цільова функція).

2.4.6 Алгоритм стохастичного градієнта

Серед проаналізованих алгоритмів оптиматизації алгоритм стохастичного градієнтного спуску виявився найефективнішим для поставленої задачі, оскільки демонструє значну швидкість роботи навіть на великому об’єму вхідних даних.

Градієнтні методи — це широкий клас оптимізаційних алгоритмів, які використовують не лише в машинному навчанні.[4] В даному випадку градієнтний підхід буде розглядатись як спосіб підбору векторів синаптичних ваг w в лінійному класифікаторі. Нехай $y^*: X \to Y$ — цільова залежність, яка відома лише на об'єктах навчальної вибірки:

$$X^l = (x_i, y_i)_{i=1}^l,\ y_i = y^*(x_i).$$

Знайдемо алгоритм $a(x,w)$, що апроксимує залежність y^*. У випадку лінійного класифікатора шуканий алгоритм має вигляд:

$$a(x,w) = \varphi\left(\sum_{j=1}^{n} w_j x^j - w_0\right),$$

де $\varphi(z)$ грає роль функції активації (в найпростішому випадку можна закласти $\varphi(z) = sign(z)$).

Згідно з принципом мінімізації емпіричного ризику, для цього достатньо вирішити оптимізаційну задачу:

$$Q(w) = \sum_{i=1}^{l} L(a(x_i, w), y_i) \to \min_w,$$

де $L(a,y)$ — задана функція втрат.

Для мінімізації використаємо метод градієнтного спуску (gradient descent). Це покроковий алгоритм, на кожній ітерації якого вектор w

змінюється в напрямку найбільшого спадання функціоналу Q (тобто в напрямку антиградієнта):

$$w := w - \eta \nabla Q(w),$$

де η — додатній параметр, який називається темпом навчання (learning rate).

Існує 2 підходи в реалізації градієнтного спуску:

- пакетний (batch), коли на кожній ітерації навчальна вибірка переглядається цілком, тільки після чого змінюється w. Це потребує великих обчислювальних затрат;
- стохастичний (stochastic/online), коли на кожній ітерації алгоритму з навчальної вибірки випадковим чином обирається лише один об'єкт. Таким чином вектор w налаштовується кожен раз на новобраний об'єкт.

Алгоритм Stochastic Gradient (SG).

Вхід:

- X^l — навчальна вибірка
- η — темп навчання
- λ — параметр згладжування функціоналу Q

Вихід:

Вектор ваг w

Тіло:

1. Ініціалізувати ваги w_j $j = 0, \dots, n$
2. Ініціалізувати поточну оцінку функціоналу:

$$Q := \sum_{i=1}^{l} L(a(x_i, w), y_i)$$

3. Повторювати:

3.1 Вибрати об'єкт x_i з X^l (наприклад, випадковим чином);

3.2 Обчислити вихідне значення алгоритму $a(x_i, w)$ та помилку:

$$\varepsilon_i := L(a(x_i, w), y_i)$$

3.3 Зробити крок градієнтного спуску:

$$w := w - \eta L'_a(a(x_i,w), y_i)\varphi'(<w,x_i>)x_i$$

3.4 Оцінити значення функціоналу:

$$Q := (1 - \lambda)Q + \lambda\varepsilon_i;$$

4. Поки значення Q не стабілізується та/або ваги w не припинять змінюватись.

Порядок вибору об'єктів.

Вище сказано, що у випадку стохастичного градієнтного спуску об'єкти слід обирати випадковим чином. Однак існують евристики, що направлені на покращення збіжності, які дещо модифікують звичайний випадковий вибір:

Перемішування (shuffling). Пропонується випадково обирати об'єкти, але поперемінно з різних класів. Ідея в тому, що об'єкти з різних класів скоріше за все менш «схожі», ніж об'єкти з одного класу, тому вектор w буде кожного разу змінюватись сильніше.

Можливий варіант алгоритму, коли вибір кожного об'єкта нерівноймовірний, при чому ймовірність випадення об'єкта обернено пропорційна величині помилки на об'єкті. Слід зауважити, що за такої евристики метод стає дуже чутливим до шумів.

Переваги SG.

Метод пристосований для динамічного (online) навчання, коли навчальні об'єкти надходять потоком, та потрібно швидко оновлювати вектор w.

Алгоритм здатен навчатись на надмірно великих вибірках за рахунок того, що випадкової підвибірки може вистачати для навчання.

Можливі різноманітні стратегії навчання. Якщо вибірка надмірно велика, або навчання відбувається динамічно, то є допустимим не зберігати навчальні об'єкти. Якщо вибірка маленька, то можна повторно подавати для навчання ті самі об'єкти.

Недоліки SG та способи боротьби з ними.

Алгоритм може не збігатись або збігатись занадто повільно. Як правило, функціонал Q має багато екстремумів та процес градієнтного спуску може «застрягти» на одному із локальних мінімумів. Для боротьби з цим використовують техніку струшування коефіцієнтів (jog of weights). Вона полягає у тому, що при кожній стабілізації функціонала робити випадкові модифікації вектора w в достатньо широкому околі поточного значення та запускати процес градієнтного спуску з нових точок.

Висновки до розділу

В розділі наведено формальну постановку досліджуваної задачі класифікації кредитів. Розглянуто вхідні та вихідні данні моделі.

Детально описанні методи машиного навчання, які можуть використовуватися для вирішення поставленної задачі. Розроблено модель, яка класифікує позики за допомогою методу зворонтнього поширення помилки з використанням алгоритму стохастичного градієнтного спуску. Результуючим класифікатором є згорткова нейронна мережа.

3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура системи підтримки прийняття рішень

Архітектура СППР визначається характером взаємодії основних її складових:

- інтерфейсу користувача;
- бази та сховища даних, документів і правил;
- моделей і аналітичних інструментів;
- інфраструктури комунікацій і мереж;
- а також елементів цих частин.

Ефективне поєднання всіх елементів СППР дає змогу уникнути ряду труднощів щодо побудови СППР і підвищити продуктивність комп'ютерної системи за рахунок: особливої інтеграції бази даних СППР з іншими внутрішніми і зовнішніми базами даних; скорочення тривалості очікування відповіді на запит користувача; ефективного використання великих математичних моделей; вдалішої координації діалогу з базою моделей та базою даних; поліпшення розуміння програмістами окремих аспектів системи; зниження витрат на створення та експлуатацію системи; мінімізування вартості підтримки та збільшення продуктивності користувачів, включаючи уникнення збоїв системи та інших проблем щодо продуктивності; зменшення інфраструктурних перешкод, які затримують розгортання нових додатків інформаційних систем і технологій, особливо СППР.

Серед факторів, які впливають на вибір конкретної архітектури СППР, можна виокремити такі: необхідність подальшого розвитку комп'ютерної системи та адаптації, зокрема, за рахунок включення в неї наявних програмних засобів; застосування еволюційного підходу до розвитку СППР. Засоби підтримки прийняття рішень мають бути так розподілені на мережах, щоб досягати до творців рішень і забезпечувати необхідний захист інформації.

Мережа є важливим елементом інфраструктури, що найбільше сприяє функціонуванню системи підтримки прийняття рішень. У сучасних СППР широко застосовуються такі головні мережеві технології:

- інтернет (Internet), який уможливлює з'єднання окремих індивідів у планетарному масштабі;
- екстранет (Extranet), що забезпечує зв'язок окремих компаній між собою.
- інтранет (Intranet), який призначений для з'єднання індивідів усередині компаній.

Складовою частиною архітектури СППР є проект мережі. Питання захисту СППР тісно пов'язані з їх архітектурою і мережевими альтернативами. Ці три теми тісно переплітаються і є дуже важливими з погляду побудови ефективної системи підтримки прийняття рішень. Якщо СППР не орієнтована на автономний (не підключений до мережі) комп'ютер у захищеному офісному середовищі, де він знаходиться під пильним оком менеджера, який його використовує, то потрібно обов'язково проводити сумісне розроблення архітектури СППР, організації мережі і розв'язувати питання стосовно захисту інформації. Можна поєднати головні компоненти СППР - інтерфейс користувача, базу даних, моделі й аналітичні інструментальні засоби, а також мережеву структуру СППР у загальну архітектуру СППР (Рис.3.1).

Рисунок 3.1 - Загальна архітектура СППР

Головним компонентом у проекті СППР є інтерфейс користувача. До інструментальних засобів для побудови інтерфейсу користувача належать: симулятори інтерфейсу, СППР-генератори, інструментальні засоби запиту і звітів, пакет розроблення кінцевого користувача (front-end). Інтерфейси користувачів СППР можуть бути безпосередньо у клієнтів за архітектури "товстого клієнта" (thick-client) або доставлені мережею, використовуючи Web-сторінки чи Java applets (Java-додатки), в архітектурі "тонкого клієнта" (thin-client). Архітектура "тонкого клієнта", де користувач взаємодіє з використанням Web-сторінок, має багато переваг, але донедавна витонченість інтерфейсу користувача була обмежена принципами архітектури "товстого клієнта", коли програма зберігається в комп’ютері користувача СППР.

3.2 Вимоги до технічного та програмного забезпечення

Мінімальна конфігурація технічних засобів серверу:

- процесор на основі команд x86, Intel Pentium 4, Pentium D, Core, AMD Athlon, Phenom, або інший CISC-процесор чи гібридний процесор;
- об’єм ОЗП: 16Гб;
- об'єм ПЗП: 1Тб;
- частота процесору: 2 ГГц;
- активне підключення до глобальної мережі;

Необхідно встановлені Microsoft SQL Server та .NET Core 2.2 Runtime & Hosting Bundle for Windows (v2.2.7)

Мінімальна конфігурація технічних засобів клієнта:

Вимоги до конфігурації технічних засобів не пред’являються. Додаток має запускатися у будь-якому сучасному веб-браузері.

3.3 Технології розробки

Оскільки WEB-сайт являється чи не найрозповсюдженішою формою програмного забезпечення у банківські сфері, для розробки системи підтримки

прийняття рішень в управлінні кредитною заборгованістю було обрано саме його. Тому потрібно вибрати мову програмування та середовище розробки, які б максимально сприяли ефективній, якісній та швидкій розробці системи. Вибираючи із існуючих технологій розробки WEB-систем, було виділено технологію ASP.NET Core.

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів.

З одного боку, ASP.NET Core є продовженням розвитку платформи ASP.NET. Але з іншого боку, це не просто черговий реліз. Вихід ASP.NET Core фактично означає революцію всієї платформи, її якісна зміна.

ASP.NET Core тепер повністю є opensource-фреймворком. Всі вихідні файли фреймворку доступні на GitHub.

ASP.NET Core може працювати поверх крос-платформної середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS X, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Хоча ASP.NET Core переважно націлене на використання .NET Core, але фреймворк також може працювати і з повною версією фреймворка .NET.

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле.

CNTK (The Microsoft Cognitive Toolki) - це уніфікований інструментарій глибокого навчання, який описує нейронні мережі як ряд обчислювальних кроків за допомогою направленого графіка. У цьому спрямованому графіку вузли аркушів представляють вхідні значення або параметри мережі, тоді як інші вузли представляють матричні операції на їх входах. CNTK дозволяє користувачам легко реалізовувати та комбінувати популярні типи моделей, такі як DNN з подачею вперед, згорткові мережі (CNN) та періодичні мережі (RNN / LSTM). Він реалізує стохастичне градієнтне спуск (SGD, зворотне розповсюдження помилок) з автоматичною диференціацією та паралелізацією на декількох графічних процесорах та серверах.

Для зберігання та опрацювання даних було обрано РСКБД Microsoft SQL Server. Серед переваг цієї РСКБД можна виділити інтеграцію з Microsoft Visual Studio та застосунками, розробленими на мовах платформи .NET.

Для розробки WEB-інтерфейсів використовується Bootstrap 4. Це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків. Репозиторій із цим фреймворком є одним із найпопулярніших на GitHub.

Microsoft Visual Studio - засіб для розробників ПЗ, яке дозволяє вирішувати основні завдання розробки: система спрощує створення, налагодження та розгортання додатків на різних платформах, включаючи SharePoint і хмарну середу. Основними перевагами Visual Studio є:

- використання обчислювальних потужностей локального комп'ютера і хмари;
- проста реалізація спільних завдань та індивідуальний підхід;
- функція підтримки декількох моніторів.

3.4 Архітектура програмного забезпечення

Програмний продукт складається з 3 модулів:

а) CreditAnalisis.Data – виконує всю роботу пов’язану з базою даних. Оперує параметрами кредиту та інформацією клієнтів. Являє собою бібліотеку класів;

б) CreditAnalisis.Execution – відповідає за логіку роботи програмного продукту. Містить в собі методи завантаження тренувальної та тестової вибірок, методи роботи з Excel, методи класифікації кредитних договорів, методи розрахунку параметрів якості кредитного портфеля (коефіцієнту покриття, коефіцієнту ліквідності, коефіцієнту забезпечення, коефіцієнту прострочених платежів, розміру резерва). Являє собою бібліотеку классів;

в) CreditAnalisis.Web – відповідає за користувацький інтерфейс та відображення даних. Керує двома іншими модулями. Представляє собою веб-додаток, який працює на IIS.

Структурна схема класів ПЗ та Entity Relationship Diagram наведені у частині графічного матеріалу.

3.5 Керівництво користувача

Користувач може зайти на сайт інтелектуальної системи для аналізу кредитного портфеля банка. Спочатку відображається головна сторінка сайту з основною інформацією про програмний продукт, його особливості та переваги. У системі працює Windows авторизація, тому у правій частині екрану користувач може побачити домен та логін. Вигляд сторінки зображено на рисунку 3.2.

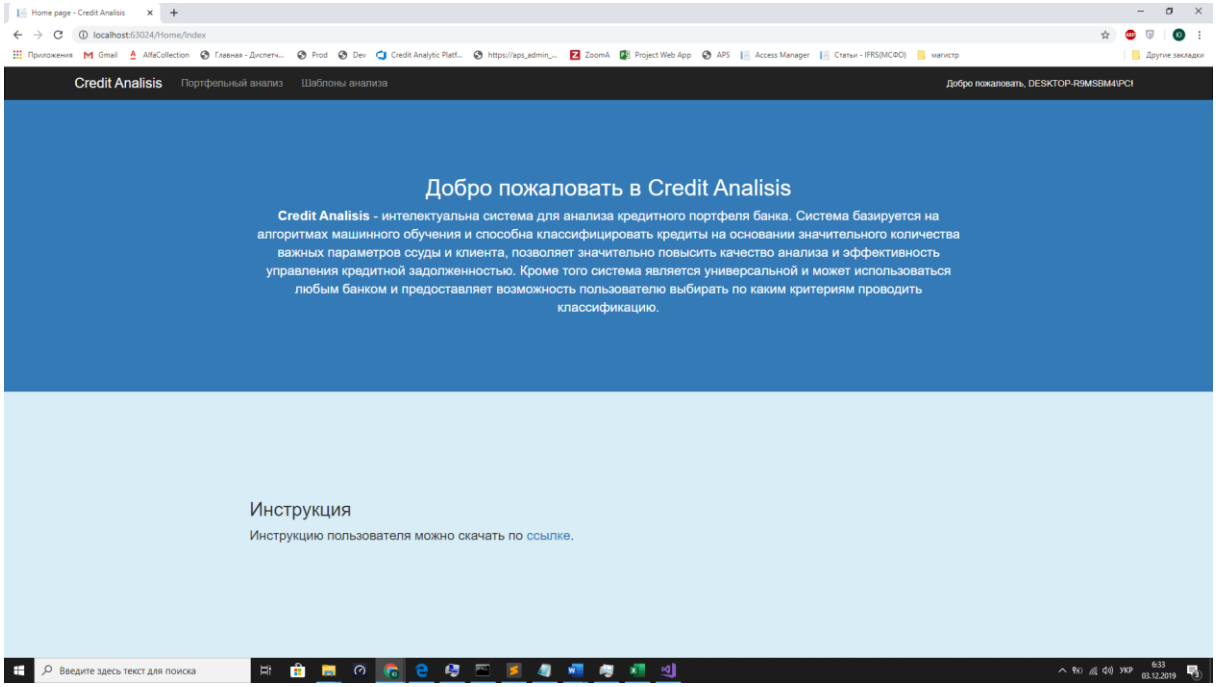


Рисунок 3.2 - Головна сторінка сайту

На вкладці «Портфельний аналіз» відображаються всі існуючі портфелі в системі. Потрібний портфель можна знайти за допомогою пошуку за назвою або описом. Вигляд сторінки зображено на рисунку 3.3.

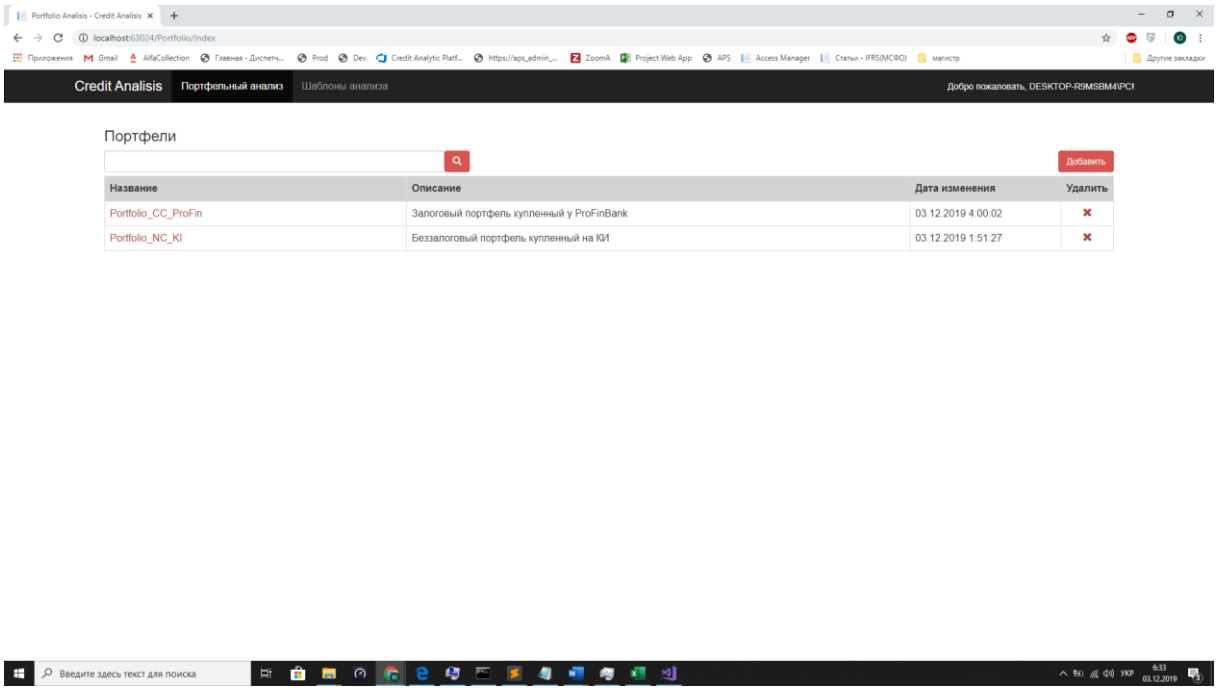


Рисунок 3.3 – Сторінка «Портфельний аналіз»

Створити новий портфель можна натиснувши кнопку «Додати». Після цього відкриється модальне вікно, на якому треба заповнити параметри нового портфеля. Вигляд модального вікна зображено на рисунку 3.4.

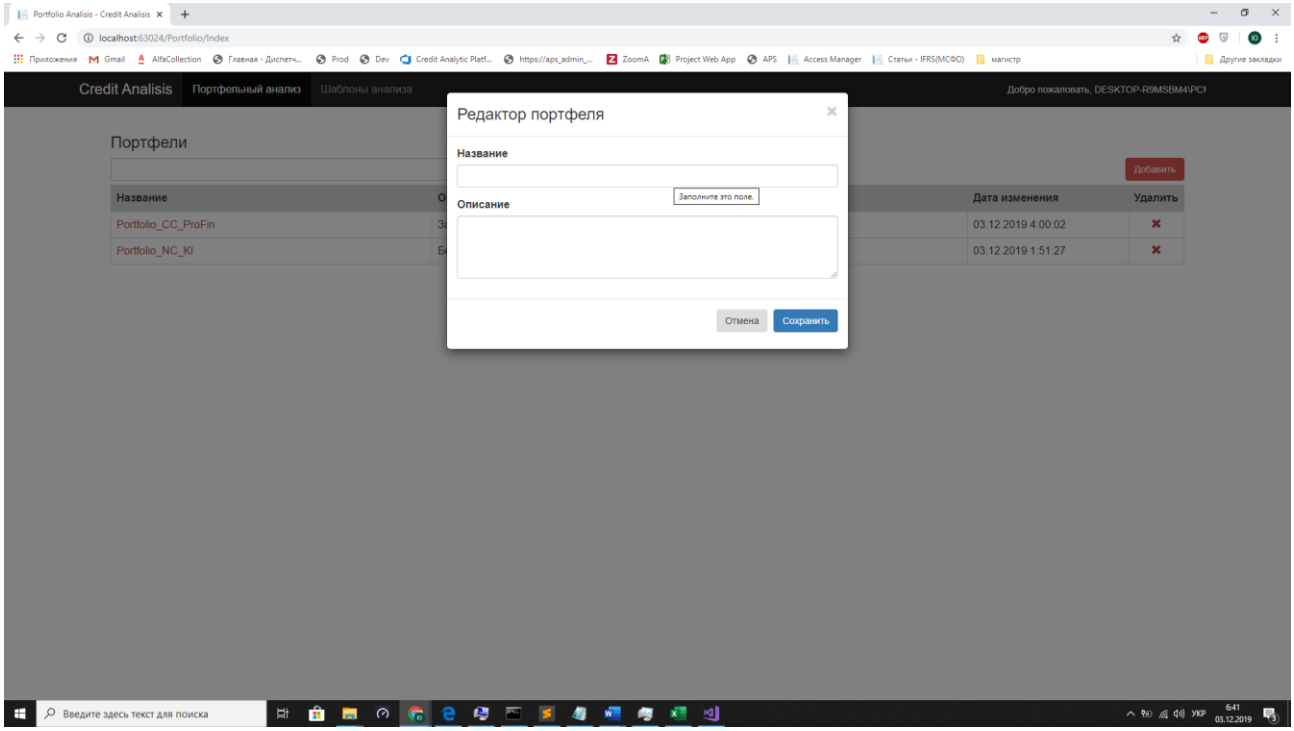


Рисунок 3.4 – Модальне вікно редагування портфелю

На сторінку портфеля можна перейти, натиснувши на його назву. На цій сторінці існує дві вкладки – «Інформація» та «Кредити». На вкладці «Інформація» відображаються параметри портфелю та результати його аналізу. Після проведення аналізу на цій вкладці можна побачити діаграми, які відображають розподіл кредитів та резервів по класам, а також коефіцієнти якості кредитного портфеля. За допомогою кнопки «Редагувати» можна редагувати параметри портфеля, кнопки «Завантажити» - завантажити кредити до портфеля з csv файла, «Експорт» - скачати кредити з портфеля у форматі csv, «Аналізувати» - провести аналіз портфеля. Вигляд вкладки «Інформація» до проведення аналізу зображено на рисунку 3.5, а після аналізу - на рисунку 3.6.

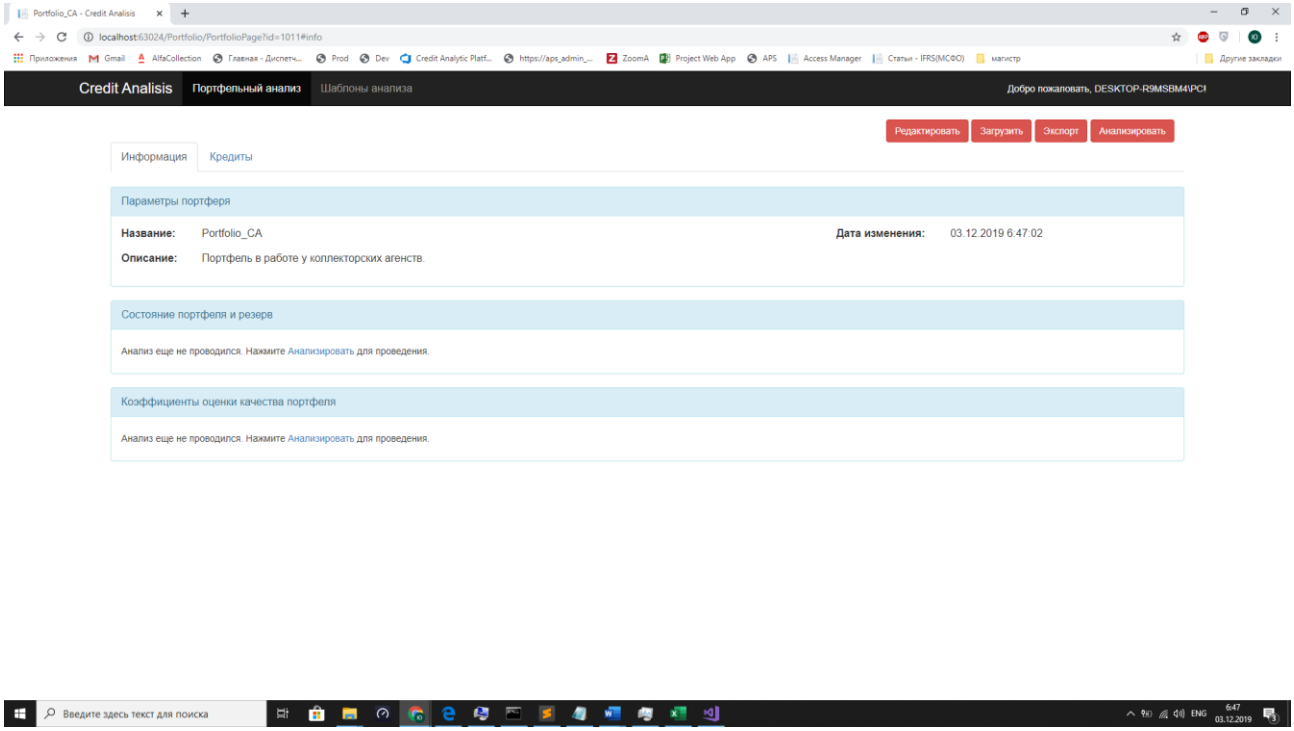


Рисунок 3.5 – Вкладка «Інформація» до проведення аналізу

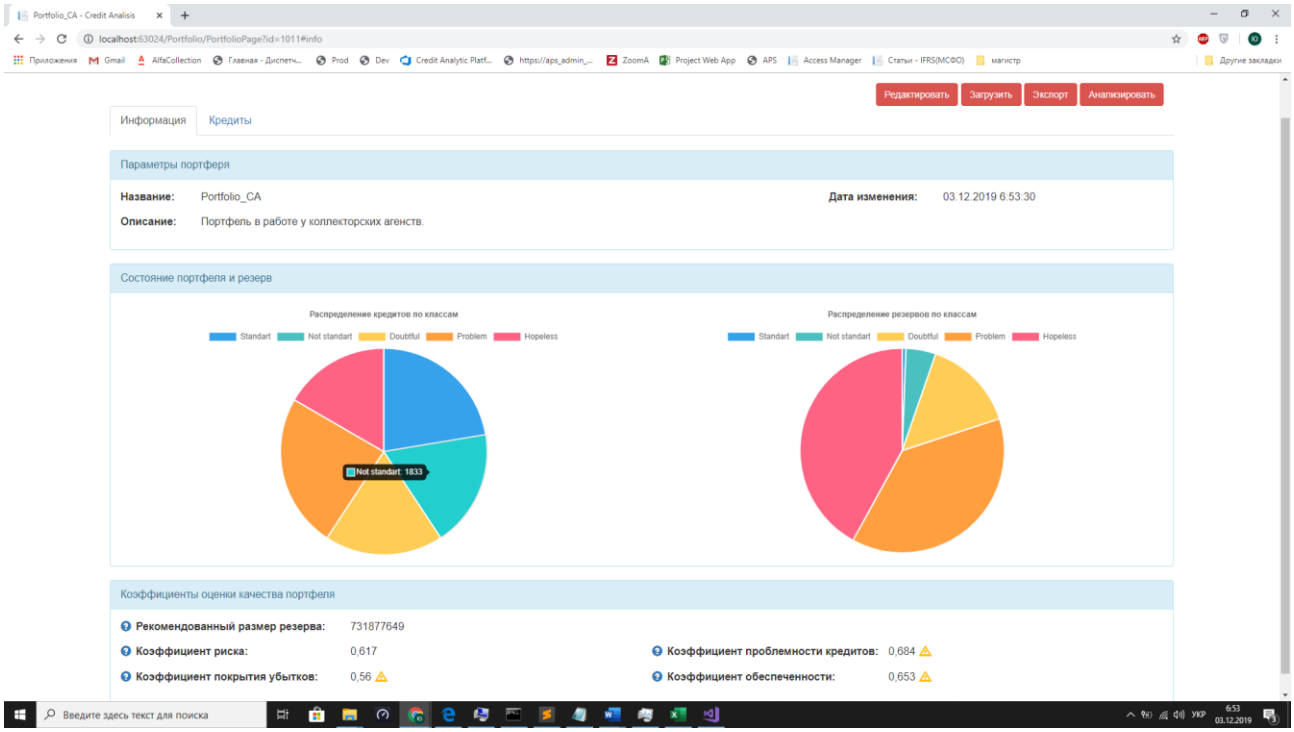


Рисунок 3.6 – Вкладка «Інформація» після проведення аналізу

На вкладці «Кредити» відображаються всі кредити завантажені до портфеля. В таблиці працює пошуг, сортування, пагінація. Вигляд сторінки зображено на рисунку 3.7.

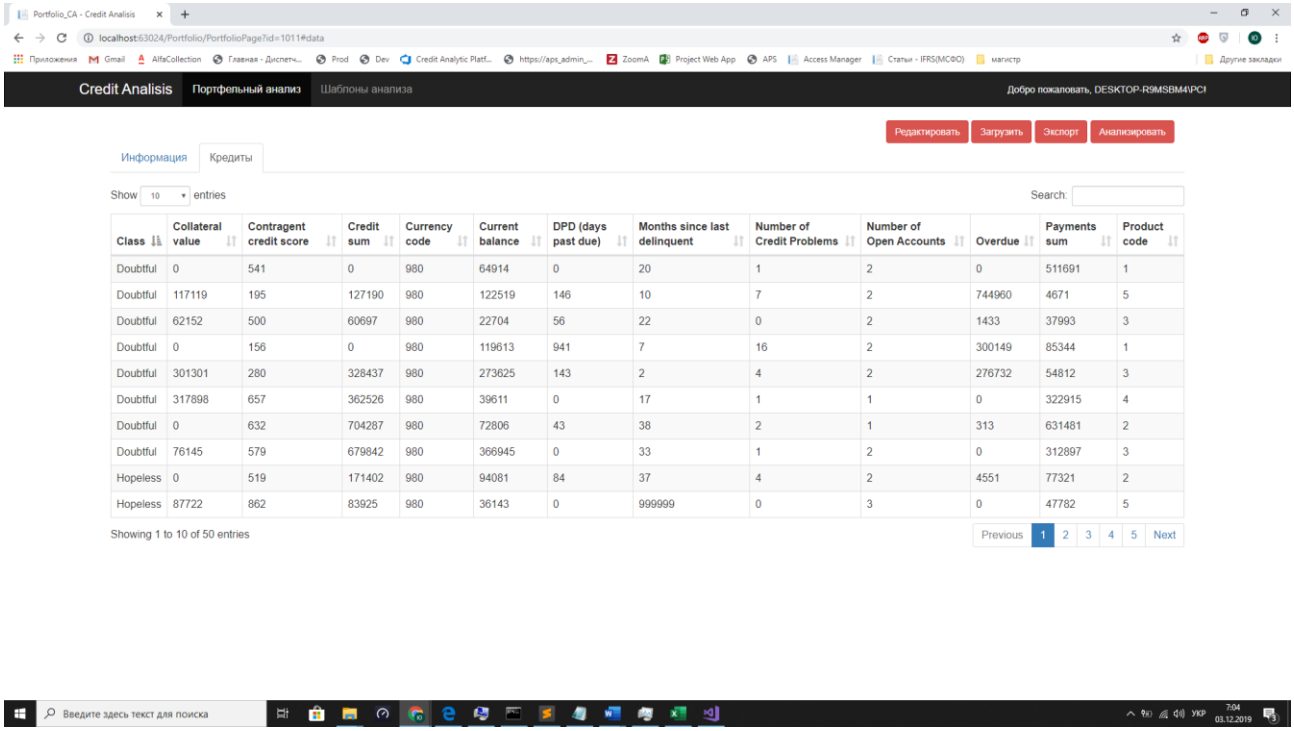


Рисунок 3.7 – Вкладка «Кредити»

В модальному вікні аналізу потрібно задати мапінг полей, які є у кредитів в портфелі та полей, які використовуються у моделі класифікації кредитів. Вигляд модального вікна зображено на рисунку 3.8.

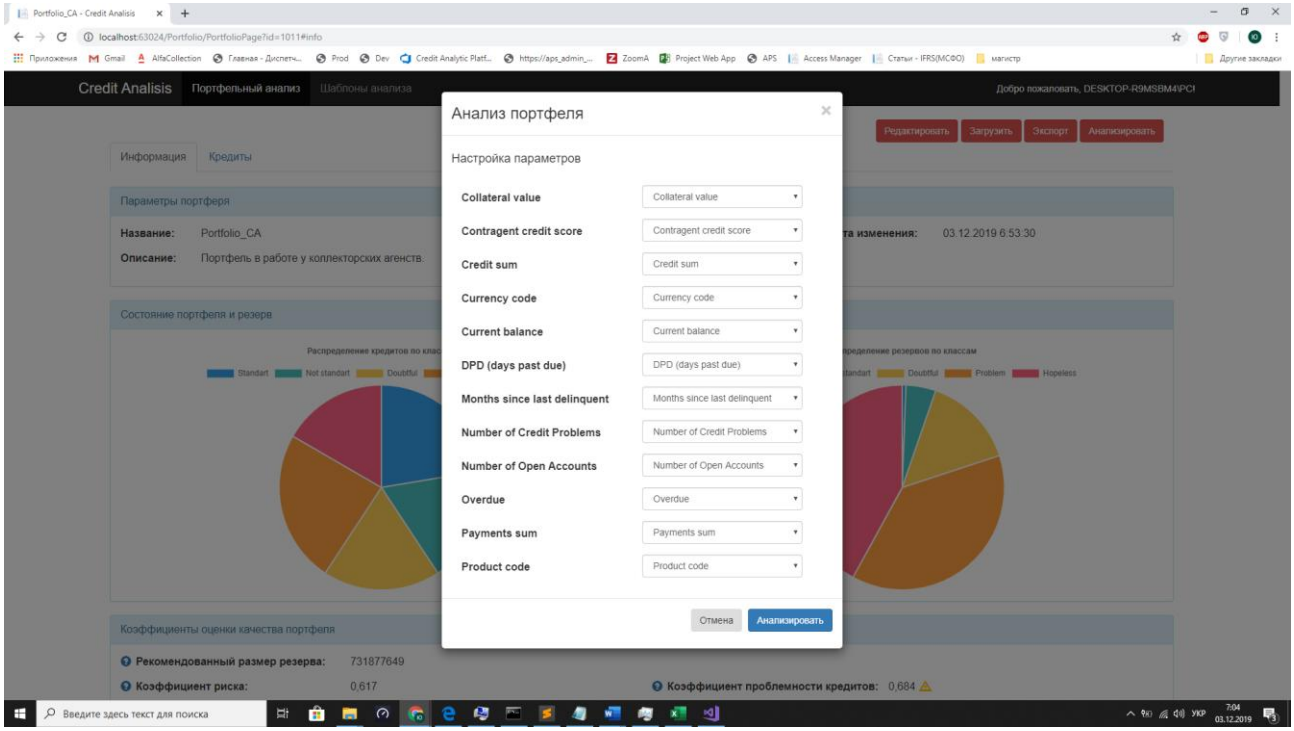


Рисунок 3.8 – Модальне вікно аналізу

Висновок до розділу

Розділ з інформаційного забезпечення присвячений програмній реалізації системи підтримки прийняття рішень для ефективного управління кредитним портфелем банку.

Детально описані засоби розробки програмного забезпечення, компоненти та архітектура СППР, вимоги до технічного та програмного забезпечення, наведена структурна схема класів, діаграма сутностей. У розділі наведено детальне керівництво користувача.

За допомогою розробленого програмного забезпечення проведено математичне моделювання аналізу кредитного портфеля банку за різними показниками. Результати моделювання, а саме побудовану нейрону мережу, було порівняно з іншими методами класифікації і наведено в таблиці 3.1.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Проведення експериментів для визначення моделі

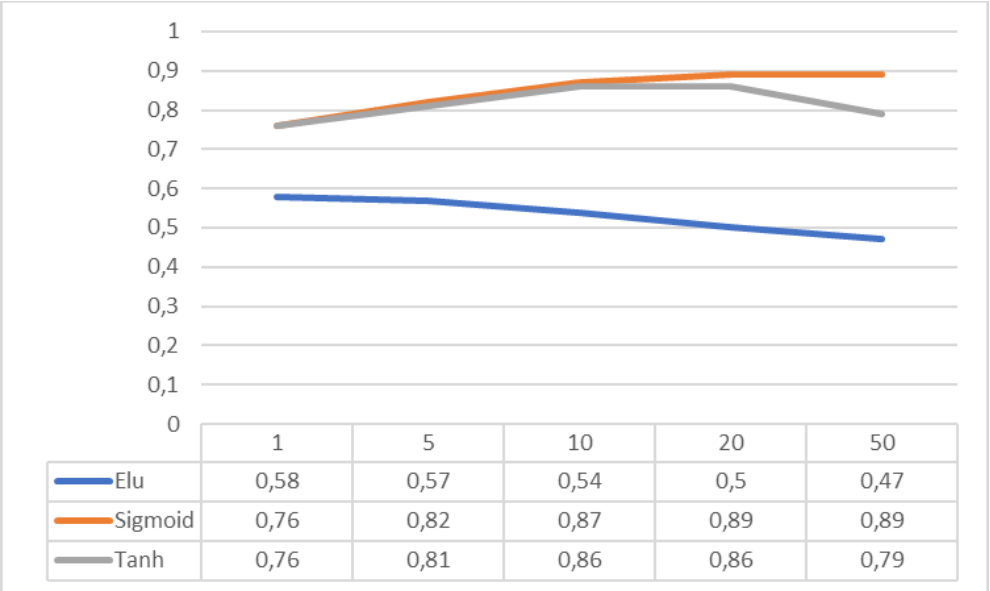
Для аналізу ефективності розробленної інформаційної технології було проведено експерименти та порівняння з іншими методами машиного навчання. Результати експериментів наведенні у таблиці 4.1.

Таблиця 4.1 – Порівняльна таблиця отриманих результатів моделей

Назва методу	Точність моделі	Якість моделі
Нейронна мережа	0,896	Дуже висока
CART (Classification and Regression Tree)	0,754	прийнятна
Мережа Байєса	0,797	висока

У даній роботі для нейронної мережі було розглянуто три функції активації: експоненцій лінійна функція, сигмоїдальна функція та гіперболічний тангенс та залежність точності классифікації від кількості епох.

Інші параметри були зафіксовані на наступних значеннях: два прихованих шари; алгоритм навчання – зворонтнього поширення помилки, швидкість навчання – 0,0001, тренувальна вибірка – 7500 кредитів, тестова - 2500.



	1	5	10	20	50
Elu	0,58	0,57	0,54	0,5	0,47
Sigmoid	0,76	0,82	0,87	0,89	0,89
Tanh	0,76	0,81	0,86	0,86	0,79

Рисунок 4.1 – Графік залежності точності розпізнавання від епохи для функцій активації

Результати роботи виявились наступними: експоненційна лінійна функція показала найгірший результат, сигмоїдальна функція та гіперболічний тангенс дають майже однаковий результат, але на більшій кількості епох сигмоїдальна працює краще. Можна помітити, що для гіперболічного тангенса та експоненційної функції зі збільшенням епох з’являється перенавчання. Для оцінки впливу норми навчання на якість класифікації було обрано три значення: 0.01, 0.001 та 0.0001. У результаті найменша норма навчання зменшує кількість помилок, тим не менш, вона потребує трохи більше часу на навчання, тому якщо немає потреби в надзвичайній точності класифікації використання занадто малої норми є надлишковою. При швидкості навчання зі значенням 0.0001 вже на першому кроці точність дорівнювала близько 75%. У свою чергу зі значеням 0.1 точність першого кроку суттєво нижча: 40%. Оптимальним можна вважати 0.0001, проте цей варіант довший і вимагає більше ресурсів.

За результатами дослідження було вирішено використовувати наступні параметри:

- функція активації – сигмоїдальна функція;
- швидкість навчання - 0.0001;

- кількість епох – 20.

Висновок до розділу.

За допомогою розробленого програмного забезпечення проведено математичне моделювання аналізу кредитного портфеля банку за різними показниками. Результати моделювання, а саме побудовану нейрону мережу, було порівняно з іншими методами класифікації і наведено в таблиці 4.1.

Була протестованва робота нейронної мережі на різних значеннях функції активації, швидкості навчання та кількості епох та обранні оптимальні параметри.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

6.1 Вступ

Стартап як форма малого ризикового (венчурного) підприємництва впродовж останнього десятиліття набула широкого розповсюдження у світі через зниження бар'єрів входу в ринок (із появою Інтернету як інструменту комунікацій та збуту стало простіше знаходити споживачів та інвесторів, займатись пошуком ресурсів, перетинати кордони між ринками різних країн), і вважається однією із наріжних складових інноваційної економіки, оскільки за рахунок мобільності, гнучкості та великої кількості стартап-проектів загальна маса інноваційних ідей зростає.

Проте створення та ринкове впровадження стартап-проектів відзначається підвищеною мірою ризику, ринково успішними стає лише невелика частка, що за різними оцінками складає від 10% до 20%. Ідея стартап-проекту, взята окремо, не вартує майже нічого: головним завданням керівника проекту на початковому етапі його існування є перетворення ідеї проекту у працюючу бізнес-модель, що починається із формування концепції товару (послуги) для визначеної клієнтської групи за наявних ринкових умов.

Розроблення та виведення стартап-проекту на ринок передбачає здійснення низки кроків, в межах яких визначають ринкові перспективи проекту, графік та принципи організації виробництва, фінансовий аналіз та аналіз ризиків і заходи з просування пропозиції для інвесторів. Узагальнено етапи розроблення стартап-проекту можна подати таким чином.

Етапи розроблення стартап-проекту:

- маркетинговий аналіз стартап-проекту. В межах цього етапу: розробляється опис самої ідеї проекту та визначаються загальні напрями використання потенційного товару чи послуги, а також їх відмінність від конкурентів, аналізуються ринкові можливості щодо його реалізації, на базі

аналізу ринкового середовища розробляється стратегія ринкового впровадження потенційного товару в межах проекту;

- організація стартап-проекту. В межах цього етапу: складається календарний план-графік реалізації стартап-проекту, розраховується потреба в основних засобах та нематеріальних активах, визначається плановий обсяг виробництва потенційного товару, на основі чого формується потреба у матеріальних ресурсах та персоналі, розраховуються загальні початкові витрати на запуск проекту та планові загальногосподарські витрати, необхідні для реалізації проекту;

- фінансово-економічний аналіз та оцінка ризиків проекту. В межах цього етапу: визначається обсяг інвестиційних витрат, розраховуються основні фінансово-економічні показники проекту (обсяг виробництва продукції, собівартість виробництва, ціна реалізації, податкове навантаження та чистий прибуток) та визначаються показники інвестиційної привабливості проекту (запас фінансової міцності, рентабельність продажів та інвестицій, період окупності проекту), визначається рівень ризикованості проекту, визначаються основні ризики проекту та шляхи їх запобігання (реагування на ризики);

- заходи з комерціалізації проекту. Цей етап спрямовано на пошук інвесторів та просування інвестиційної пропозиції (оферти). Він передбачає: визначення цільової групи інвесторів та опису їх ділових інтересів, складання інвест-пропозиції (оферти): стислої характеристики проекту для попереднього ознайомлення інвестора із проектом, планування заходів з просування оферти: визначення комунікаційних каналів та площадок та планування системи заходів з просування в межах обраних каналів, планування ресурсів для реалізації заходів з просування оферти.

6.2 Опис ідеї проекту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

− зміст ідеї (що пропонується);

− можливі напрямки застосування;

− основні вигоди, що може отримати користувач товару (за кожним напрямом застосування);

− чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подані у вигляді таблиці (таблиця 5.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрями застосування	Вигоди для користувача
Розробка системи, яка дозволяє виконувати аналіз кредитного портфеля банку.	Аналіз фінансового стану і рівня забезпеченості позичальника та рівня обслуговування кредиту.	Дозволяє своєчасно та повно оцінити стан окремого кредита та портфеля в цілому.
	Виділення корисної інформації серед великих масивів даних про кредит та позичальника.	Дозволяє будувати классифікуючу модель на основі значущих змінних.
	Виявлення можливості виникнення проблем з повернення коштів позичальником на ранньому етапі.	Дозволяє знаходити (попереджати) проблеми, усувати їх на ранніх етапах, коли ціна помилки, не сильно велика. Допомагає скоригувати стратегію просування продукту на ринок.
	Автоматизація систем моніторингу (відстеження змін у динаміці).	Дає змогу зрозуміти тренди у ефективності менеджменту кредитного портфеля.

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

− визначення переліку техніко-економічних властивостей та характеристик ідеї;

− визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проведення збір інформації щодо значень техніко економічних показників для ідеї власного проекту та проектів конкурентів відповідно до визначеного вище переліку;

− проведення порівняльного аналізу показників: для власної ідеї визначені показники, що мають гірші значення (W, слабкі), аналогічні значення (N, нейтральні), кращі значення (S, сильні) (таблиця 5.2).

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	Мій проект	Кон-нт 1	Кон-нт 2	Кон-нт 3			
Форма виконання	Веб додаток	Веб додаток	Веб додаток	Програма		+	
Собівартість	Низька	Середня	Низька	Висока			+
Наявність адміністратора	Не треба, дистанційно	Треба	Треба	Треба			+
Наявність інтернету	Не треба	Треба	Треба	Не треба			+
Кросплатформенність	Так	Так	Так	Ні		+	
Складність використа-	Так	Ні	Так	Ні		+	

ння							

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

6.3 Технологічний аудит ідеї проекту

В межах даного підрозділу проведено аудит технології, за допомогою якої реалізовано ідею проекту.

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 5.3):

- за якою технологією буде виготовлено товар згідно ідеї проекту?
- чи існують такі технології, чи їх потрібно розробити/доробити?
- чи доступні такі технології авторам проекту?

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
Створення програмного забезпечення	Visual Studio	Наявна	Безкоштовна, доступна
	.NET.Core	Наявна	Безкоштовна, доступна
	CNTK	Наявна	Безкоштовна, доступна
	Bootstrap 4	Наявна	Безкоштовна, доступна

Обрана технологія реалізації ідеї проекту: для створення додатку обрана технологія, яка є безкоштовною та якою володіють розробники.

6.4 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	5000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект. За результатами аналізу таблиці

зроблено висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням. Так, є. Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 5.5).

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Необхідне програмне забезпечення, яке здатне класифікувати кредити та робити аналіз кредитного портфеля банку.	Потенційними цільовими групами є банки, великі компанії, організації, які займються наданням кредитів, компанії, агенства які займаються стягненням заборгованності.	Цільова аудиторія має достатньо великі масиви даних про фінансовий стан кредиту та його історію, позичальників і історію їх виплати, аналіз яких може надати великі переваги при керуванні кредитним портфелем.	Рішення має бути швидким, зрозумілим у використанні (інтуїтивно зрозумілим), не бути сильно дорогим, без нагромадження зайвого функціоналу. Використання даного продукту вивільнить час на більш важливі і складні задачі.

Після визначення потенційних груп клієнтів проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 5.6 — 5.7). Фактори в таблиці подані в порядку зменшення значущості.

Таблиця 5.6 – Фактори загроз

	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку; 2) Запропонувати великій компанії поглинути себе; 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок.
2	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом (потрібний аналіз дебітного портфелю, а не кредитного)	Передбачити можливість додавання нового функціоналу до створюваного ПЗ.

Таблиця 5.7 – Фактори можливостей

	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання можливостей потенційних покупців.	Ріст зацікавленості до продукту серед інших груп користувачів.	Додати підказки, інструкції та демонстрації роботи системи.
2	Зниження довіри до конкурента 2.	У ПЗ конкурента №3 нещодавно була знайдена помилка.	При виході на ринок звертати увагу покупців на безпеку нашого ПЗ.

У таблиці 5.8 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Вказати тип конкуренції — досконала	Існує 3 фірми конкурента на ринку	Врахувати ціни конкурентних компаній на початкових етапах , вказати на конкретні переваги перед конкурентами
За галузевою ознакою — внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Створити основу інфраструктури таким чином, щоб можна було легко переробити дану інфраструктуру для використання у інших галузях
Конкуренція за видами товарів — товарно-видова	Види товарів є однаковими, а саме — послуга створення	Створити інфраструктуру, враховуючи недоліки конкурентів
За характером конкурентних переваг — нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти

У таблиці 5.9 наведено аналіз конкуренції в галузі за М. Портером.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори и сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 3 — його рішення представлене у вигляді ПЗ	Так, можливості для входу на ринок є, бо наше рішення покращує та пришвидшує роботу спеціаліста.	Постачальники відсутні	Важливим для користувача є кросплатформеність ПЗ та якість його роботи	Товари-замінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару

За результатами аналізу таблиці зроблено висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також зроблено висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності.

На основі аналізу конкуренції, проведеного в таблиці 5.9, а також із урахуванням характеристик ідеї проекту (таблиця 5.2), вимог споживачів до товару (таблиця 5.5) та факторів маркетингового середовища (таблиці 5.6 – 5.7) визначено та обґрунтовано перелік факторів конкурентоспроможності. Аналіз оформляється за таблицею 5.10.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
Виконання програмного забезпечення у кросплатформеному вигляді	Можливість використання програмного забезпечення на будь-якій платформі.
Ціна	Дане програмне рішення має невисоку ціну

За визначеними факторами конкурентоспроможності (таблиця 5.10) проведено аналіз сильних та слабких сторін стартап-проекту (таблиця 5.11).

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

Фактор конкуренто-спроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
		-3	-2	-1	0	1	2	3
Кросплатфор-менне програмне забезпечення	17			+				
Ціна	20		+					

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 5.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 5.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони: ціна, кросплатформеність	Слабкі сторони: у деяких випадках можуть стомлюватися руки
Можливості: більш широке розповсюдження технологій з підтримкою віртуальної реальності, поява нових технологій моніторингу навколишнього середовища.	Загрози: видавлювання з ринку конкурентами, зміна потреб користувачів.

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. таблицю 5.9, аналіз потенційних конкурентів). Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 5.13).

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розробка програмного продукту, PR, просування бренду	90%	9 місяців
2	Розробка програмного продукту, безкоштовне розповсюдження	60%	6 місяців

6.5 Розроблення ринкової стратегії стартапу

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 5.14).

Таблиця 5.14 - Опис цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів прийняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Користувачі смартфонів з віком 10 – 40	Висока	Високий	Висока	Середня
Користувачі	Невисока	Невисока	Низька	Середня

смартфонів з ОС				

За результатами аналізу потенційних груп споживачів було обрано цільову групу, для якої буде запропоновано даний товар, та визначено стратегію охоплення ринку — стратегію концентрованого маркетингу (компанія зосереджується на одному сегменті). Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиці 5.15 – 5.17).

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розробка програмного продукту, PR, просування бренду	Масовий маркетинг	Екстрановий спосіб рухової взаємодії, який не потребує громіздких додаткових девайсів	Стратегія диференціації

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопроходцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Забирати існуючих	Ні	Стратегія наслідування

Таблиця 5.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Невисока ціна	Позиціювання за показниками ціни	Ціна	Ціна, зручність, швидкість роботи

Висновки до розділу

В результаті виконання даного розділу магістерської дисертації було з’ясовано, що існує реальна можливість ринкової комерціалізації розробленого продукту. Також слід зазначити, що даний продукт буде рентабельним, оскільки ліцензії на продукти-аналоги мають високу ціну.

З огляду на потенційні групи клієнтів та бар’єри, які стоять на шляху, можна сказати, що у даного проекту є досить непогані перспективи впровадження. При цьому рівень конкуренції на даний момент є не дуже високим, а конкурентоспроможність проекту є достатньою.

ВИСНОВКИ

При виконанні магістерської дисертації були опрацьовані теоретичні та методологічні основи оцінки якості кредитного портфеля, був проведений огляд сучасних підходів до аналізу кредитного портфеля, були розглянуті статистичні методи, методи машиного навчання, а саме: згорткові нейронні мережі.

Наведена формальна постановка задачі класифікації кредитів, здійснено математичне моделювання, створений алгоритм, що включає в себе класифікацію за великим спектром ознак, використовуючи згорткову нейронну мережу.

Проаналізована робота створенної нейронної мережі з різними параметрами, наведені графіки залежності точності класифікації від функції активації, швидкості навчання та кількості епох. За результатами дослідження обрана оптимальна модель.

Створено систему підтримки прийняття рішень для ефективного управління кредитним портфелем, що базується на алгоритмах машинного навчання та здатна класифікувати кредити на основі значної кількості важливих параметрів позики та клієнта, яка дозволяє значно підвищити якість аналізу та ефективність управління кредитною заборгованністю.

Проведено дослідження ефективності розробленної інформаційної технології та порівняльний аналіз з існуючими.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Соколинская Н.Э. Проблемы менеджмента кредитного портфеля в современных условиях - М.: Банковское дело, 2011. -155 с.

2) Кроливецкая Л.П., Тихомирова Е.В. Банковское дело: Кредитная деятельность коммерческих банков - М.: КНОРУС, 2009. - 280 с.

3) Зайченко Ю.П. Основы проектирования интеллектуальных систем. - Киев: Слово, 2004. –352 с.

4) Николенко С.И. Deep learning II: Градиентный спуск. – НИУ ВШЭ − Санкт-Петербург, 2017. – 3 с.

5) Пірог В. В. Оцінювання якості кредитного портфеля банків з урахуванням виконання економічних нормативів НБУ / В. В. Пірог // Науковий вісник НЛТУ України. — 2011. — Вип. 21.18. — С. 228–235.

6) Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development 3 (3): 210–229.

7) Ясницкий, Л. Н. Введение в искусственный интеллект. с. 70–72.

8) T. Xiao, J. Zhang et al., “Error-driven incremental learning in deep convolutional neural network for large-scale image classification,” in International Conference on Multimedia, no. 22. ACM, 2014, pp. 177–186.

9) Барцев С. И., Гилев С. Е., Охонин В. А., Принцип двойственности в организации адаптивных сетей обработки информации, В кн.: Динамика химических и биологических систем. — Новосибирск: Наука, 1989. — С. 6-55.

10) Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

11) Офіційний сайт Національного банку України. – [Електронний ресурс]. – Режим доступу: https://bank.gov.ua/control/uk/index.

12) Аванесова Н.Е. Кредитна політика банківських установ: стратегічні аспекти та ризики / Н.Е. Аванесова, Ю.М. Вознікова // Глобальні та національні проблеми економіки. – 2015. – № 8. – С. 956-961.

13) Дзюблюк, О. Механізм забезпечення якості кредитного портфеля й управління кредитним ризиком банку в період кризових явищ в економіці [Текст] / О.Дзюблюк //Журнал європейської економіки - 2010. - № 1. - С.108-125.

14) Банківські операції: Підручник / А. М. Мороз, М. І. Савлук, М. Ф. Пуховкін та ін. / За ред. А. М. Мороза. — К.: КНЕУ, 2000. — 384 с.

15) Голуб, В.М. Управління кредитним портфелем комерційного банку [Текст]: Автореф. дис. канд.екон. наук: 08.04.01 / В.М. Голуб; Київ. нац. екон. ун-т. – К., 2004. – 19 с.

16) Положення про визначення банками України розміру кредитного ризику за активними банківськими операціями// Постанова Правління Національного банку України 30.06.2016 № 351 – [Електронний ресурс]. – Режим доступу: http://zakon3.rada.gov.ua/laws/show/v0351500-16.

ДОДАТОК А ОПИС ПРОГРАМИ

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Text;

namespace CreditAnalisis.Data.Domain
{
    public class Credit
    {
        public int Id { get; set; }
        public int PortfolioId { get; set; }
        public int ClassId { get; set; }

        public Credit() { }
        public Credit(IDataReader dataReader)
        {

        }
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;

namespace CreditAnalisis.Data.Domain
{
    public class CreditClass
    {
        public int? Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int? CreditCount { get; set; }
        public int? CreditReserveAmount { get; set; }
        public int? ReservedPerc { get; set; }
```

```
            public CreditClass() { }
            public CreditClass(IDataReader reader)
            {
                var cols = reader.GetSchemaTable().Rows.Cast<DataRow>().Select(row =>
row["ColumnName"] as string).ToList();
                Id = (int)reader["id"];
                Name = reader["name"].ToString();
                Description = reader["Description"].ToString();
                CreditCount = cols.Contains("CreditCount") ?
(int?)reader["CreditCount"] : null;
                CreditReserveAmount = cols.Contains("CreditReserveAmount") ?
(int?)reader["CreditReserveAmount"] : null;
                ReservedPerc = cols.Contains("ReservedPercentage") ?
(int?)reader["ReservedPercentage"] : null;
            }
        }
    }
    using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Text;

    namespace CreditAnalisis.Data.Domain
    {
        public class CreditProperty
        {
            public int Id { get; set; }
            public string Value { get; set; }
            public int PropertyId { get; set; }
            public int CreditId { get; set; }

            public CreditProperty() { }
            public CreditProperty(IDataReader dataReader)
            {

            }
        }
    }
    using System;
    using System.Collections.Generic;
```

```
using System.Data;
using System.Text;

namespace CreditAnalisis.Data.Domain
{
    public class Criterion
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int PatternId { get; set; }
        public int? PropertyId { get; set; }

        public Criterion() { }
        public Criterion(IDataReader reader)
        {
            Id = (int)reader["id"];
            Name = reader["name"].ToString();
            Description = reader["description"].ToString();
            PatternId = (int)reader["patternId"];
        }
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Text;

namespace CreditAnalisis.Data.Domain
{
    public class Pattern
    {
        public int? Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime UpdateDate { get; set; }

        public Pattern() { }
        public Pattern(IDataReader reader)
        {
```

```
                Id = (int)reader["id"];
                Name = reader["name"].ToString();
                Description = reader["Description"].ToString();
                UpdateDate = reader.GetDateTime(reader.GetOrdinal("UpdatedAt"));
            }
        }
    }
    using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Text;

    namespace CreditAnalisis.Data.Domain
    {
        public class Portfolio
        {
            public int? Id { get; set; }
            public string Name { get; set; }
            public string Description { get; set; }
            public bool IsAnalyzed { get; set; }
            public DateTime UpdateDate { get; set; }
            public int? ReservedSum { get; set; }
            public double? CoverageCofficient { get; set; }
            public double? ProvisionCofficient { get; set; }
            public double? RiskCoefficient { get; set; }
            public double? OverduePaymentCofficient { get; set; }

            public Portfolio() { }
            public Portfolio(IDataReader reader)
            {
                Id = (int) reader["id"];
                Name = reader["name"].ToString();
                Description = reader["Description"].ToString();
                UpdateDate = reader.GetDateTime(reader.GetOrdinal("UpdatedAt"));
                IsAnalyzed = (int)reader["IsAnalized"] == 1;
                ReservedSum = reader["ReservedSum"] == DBNull.Value ? ReservedSum :
(int) reader["ReservedSum"];
                CoverageCofficient = reader["CoverageCofficient"] == DBNull.Value ?
CoverageCofficient : (double)reader["CoverageCofficient"];
```

```
                ProvisionCofficient = reader["ProvisionCofficient"] == DBNull.Value ?
ProvisionCofficient : (double)reader["ProvisionCofficient"];
                RiskCoefficient = reader["RiskCoefficient"] == DBNull.Value ?
RiskCoefficient : (double)reader["RiskCoefficient"];
                OverduePaymentCofficient = reader["OverduePaymentCofficient"] ==
DBNull.Value ? OverduePaymentCofficient : (double)reader["OverduePaymentCofficient"];
            }
        }
    }
    using System;
    using System.Collections.Generic;
    using System.Data;
    using System.Text;

    namespace CreditAnalisis.Data.Domain
    {
        public class Property
        {
            public int Id { get; set; }
            public string Name { get; set; }
            public string Description { get; set; }
            public int PortfolioId { get; set; }
            public int CriterionId { get; set; }

            public Property() { }
            public Property(IDataReader reader)
            {
                Id = (int)reader["id"];
                Name = reader["name"].ToString();
                Description = reader["description"].ToString();
                PortfolioId = (int)reader["portfolioId"];
                if (reader["criterionId"] != null && reader["criterionId"].ToString()
!= "") CriterionId = (int)reader["criterionId"];
            }
        }
    }
    using CreditAnalisis.Data.Domain;
    using System;
    using System.Collections.Generic;
    using System.Data;
```

```
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;

namespace CreditAnalisis.Data.Services
{
    public class PatternService
    {
        string connectionString;

        public PatternService(string connectionString)
        {
            this.connectionString = connectionString;
            this.connectionString = "Data Source=localhost;Initial
Catalog=CreditAnalisis;Integrated Security=True;Pooling=False";
        }

        public List<Criterion> GetCriterions(int patternId)
        {
            var criterions = new List<Criterion>();
            try
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand(string.Format("SELECT * FROM
Criterion WHERE patternId = {0} ORDER BY Name", patternId), conn);
                command.CommandType = CommandType.Text;
                var reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read()) // построчно считываем данные
                    {
                        criterions.Add(new Criterion(reader));
                    }
                }
                reader.Close();
                command.Dispose();
                conn.Close();
            }
```

```
            catch (Exception e)
            {
                //TODO
            }
            return criterions;
        }

        public List<CreditClass> GetClasses(int patternId)
        {
            var classes = new List<CreditClass>();
            try
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand("SELECT * FROM Class ORDER BY Id",
conn);
                command.CommandType = CommandType.Text;
                var reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read()) // построчно считываем данные
                    {
                        classes.Add(new CreditClass(reader));
                    }
                }
                reader.Close();
                command.Dispose();
                conn.Close();
            }
            catch (Exception e)
            {
                //TODO
            }
            return classes;
        }

        public int SavePattern(Pattern pattern, string user)
        {
            var conn = new SqlConnection(connectionString);
            conn.Open();
```

```
                ClearCriterion(pattern.Id, conn);
                var command = new SqlCommand("SavePattern", conn);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@id", pattern.Id);
                command.Parameters.AddWithValue("@name", pattern.Name);
                command.Parameters.AddWithValue("@description", pattern.Description);
                command.Parameters.AddWithValue("@user", user);
                var returnParameter = command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int)returnParameter.Value;
                command.Dispose();
                conn.Close();
                return id;
            }

            public void DeletePattern(int id)
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand(
                   string.Format("DELETE Pattern WHERE Id = {0}", id)
                   , conn);
                command.CommandType = System.Data.CommandType.Text;
                command.ExecuteNonQuery();
                command.Dispose();
                conn.Close();
            }

            public List<Pattern> GetPatterns()
            {
                var portfolios = new List<Pattern>();
                try
                {
                    var conn = new SqlConnection(connectionString);
                    conn.Open();
                    var command = new SqlCommand("SELECT * FROM Pattern ORDER BY
Name", conn);
                    command.CommandType = CommandType.Text;
```

```
                    var reader = command.ExecuteReader();
                    if (reader.HasRows)
                    {
                        while (reader.Read()) // построчно считываем данные
                        {
                            portfolios.Add(new Pattern(reader));
                        }
                    }
                    reader.Close();
                    command.Dispose();
                    conn.Close();
                }
                catch (Exception e)
                {
                    //TODO
                }
                return portfolios;
            }

            public void ClearCriterion(int? patternId, SqlConnection conn)
            {
                if (patternId == null)
                    return;
                var command = new SqlCommand(
                    string.Format("DELETE Criterion WHERE PatternId = {0}",
patternId)
                    , conn);
                command.CommandType = System.Data.CommandType.Text;
                command.ExecuteNonQuery();
                command.Dispose();
            }

            public int CreateCriterion(Criterion criterion, SqlConnection conn)
            {
                var command = new SqlCommand("CreateCriterion", conn);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@name", criterion.Name);
                command.Parameters.AddWithValue("@description",
criterion.Description);
                command.Parameters.AddWithValue("@patternId", criterion.PatternId);
```

```
                //command.Parameters.AddWithValue("@criterionId",
property.CriterionId);
                var returnParameter = command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int)returnParameter.Value;
                command.Dispose();
                return id;
            }

            public void LoadPattern(string filename, int patternId)
            {
                try
                {
                    var conn = new SqlConnection(connectionString);
                    conn.Open();
                    try
                    {
                        ClearCriterion(patternId, conn);
                        string data = "";
                        using (var stream = new FileStream(filename, FileMode.Open,
FileAccess.Read))
                        using (var reader = new StreamReader(stream))
                        {
                            data = reader.ReadToEnd();
                        }
                        var dataList = data.Split(Environment.NewLine).ToList();

                        var properties = new List<int>();
                        foreach (var cell in dataList[0].Split(";").Skip(1))
                        {
                            CreateCriterion(new Criterion() { Name = cell, PatternId
= patternId }, conn);
                        }
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine(e.Message);
                    }
```

```
                    conn.Close();
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                }
            }
        }
    }
    using CreditAnalisis.Data.Domain;
    using System;
    using System.Collections.Generic;
    using System.Text;
    using System.Data.SqlClient;
    using System.Data;
    using NPOI.XSSF.UserModel;
    using System.IO;
    using NPOI.SS.UserModel;
    using System.Linq;
    using CreditAnalisis.Execution;
    using Newtonsoft.Json.Linq;

    namespace CreditAnalisis.Data.Services
    {
        public class PortfolioService
        {
            string connectionString;

            public PortfolioService(string connectionString)
            {
                this.connectionString = connectionString;
                this.connectionString      =      "Data      Source=localhost;Initial
Catalog=CreditAnalisis;Integrated Security=True;Pooling=False";
            }

            public int SavePortfolio(Portfolio portfolio, string user)
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand("SavePortfolio", conn);
```

```
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@id", portfolio.Id);
                command.Parameters.AddWithValue("@name", portfolio.Name);
                command.Parameters.AddWithValue("@description",
portfolio.Description);
                command.Parameters.AddWithValue("@user",  user);
                var     returnParameter     =     command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int) returnParameter.Value;
                command.Dispose();
                conn.Close();
                return id;
            }

            public void DeletePortfolio(int id)
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var transaction = conn.BeginTransaction();
                ClearPortfolioData(id, conn, transaction);
                var command = new SqlCommand("DeletePortfolio", conn, transaction);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@portfolioId", id);
                command.ExecuteNonQuery();
                command.Dispose();
                transaction.Commit();
                transaction.Dispose();
                conn.Close();
            }

            public   int   CreateProperty(Property   property,   SqlConnection   conn,
SqlTransaction transaction)
            {
                var command = new SqlCommand("CreateProperty", conn, transaction);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@name", property.Name);
                command.Parameters.AddWithValue("@description",
property.Description);
```

```
                command.Parameters.AddWithValue("@portfolioId",
property.PortfolioId);
                //command.Parameters.AddWithValue("@criterionId",
property.CriterionId);
                var     returnParameter     =     command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int)returnParameter.Value;
                command.Dispose();
                return id;
            }

            public int CreateCredit(Credit credit, SqlConnection conn, SqlTransaction
transaction)
            {
                var command = new SqlCommand("CreateCredit", conn, transaction);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@classId", credit.ClassId);
                command.Parameters.AddWithValue("@portfolioId", credit.PortfolioId);
                var     returnParameter     =     command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int)returnParameter.Value;
                command.Dispose();
                return id;
            }

            public     int     CreateCreditProperty(CreditProperty     creditProperty,
SqlConnection conn, SqlTransaction transaction)
            {
                var   command   =   new   SqlCommand("CreateCreditProperty",   conn,
transaction);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@propertyId",
creditProperty.PropertyId);
                command.Parameters.AddWithValue("@creditId",
creditProperty.CreditId);
                command.Parameters.AddWithValue("@value", creditProperty.Value);
```

```
                var returnParameter = command.Parameters.Add("@ReturnVal",
SqlDbType.Int);
                returnParameter.Direction = ParameterDirection.ReturnValue;
                command.ExecuteNonQuery();
                var id = (int)returnParameter.Value;
                command.Dispose();
                return id;
            }

            public void ClearPortfolioData(int portfolioId, SqlConnection conn,
SqlTransaction transaction)
            {
                var command = new SqlCommand("ClearPortfolioData", conn,
transaction);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@portfolioId", portfolioId);
                command.ExecuteNonQuery();
                command.Dispose();
            }

            public List<CreditClass> GetPortfolioClassInfos(int portfolioId)
            {
                var portfolioClassInfos = new List<CreditClass>();
                try
                {
                    var conn = new SqlConnection(connectionString);
                    conn.Open();
                    var command = new SqlCommand("GetPortfolioClassInfos", conn);
                    command.CommandType = CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@portfolioId", portfolioId);
                    var reader = command.ExecuteReader();
                    if (reader.HasRows)
                    {
                        while (reader.Read()) // построчно считываем данные
                        {
                            portfolioClassInfos.Add(new CreditClass(reader));
                        }
                    }
                    reader.Close();
                    command.Dispose();
```

```
                    conn.Close();
                }
                catch (Exception e)
                {
                    //TODO
                }
                return portfolioClassInfos;
            }

            public void LoadPortfolio(string filename, int portfolioId)
            {
                try
                {
                    var conn = new SqlConnection(connectionString);
                    conn.Open();
                    var transaction = conn.BeginTransaction();
                    try
                    {
                        ClearPortfolioData(portfolioId, conn, transaction);
                        string data = "";
                        using (var stream = new FileStream(filename, FileMode.Open,
FileAccess.Read))
                        using (var reader = new StreamReader(stream))
                        {
                            data = reader.ReadToEnd();
                        }
                        var dataList = data.Split(Environment.NewLine).ToList();

                        var properties = new List<int>();
                        foreach (var cell in dataList[0].Split(";"))
                        {
                            properties.Add(CreateProperty(new Property() { Name =
cell, PortfolioId = portfolioId }, conn, transaction));
                        }
                        dataList.RemoveAt(0);
                        foreach (var row in dataList)
                        {
                            var creditId = CreateCredit(new Credit() { PortfolioId =
portfolioId }, conn, transaction);
```

```
                            foreach (var cell in row.Split(";").Select((value, index)
=> new { index, value }))
                            {
                                CreateCreditProperty(new CreditProperty() { CreditId
= creditId, PropertyId = properties[cell.index],
                                        Value = cell.value }, conn, transaction);
                            }
                        }
                        transaction.Commit();

                    }
                    catch (Exception e)
                    {
                        Console.WriteLine(e.Message);
                        transaction.Rollback();
                    }
                    conn.Close();
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                }
            }

            public List<Portfolio> GetPortfolios()
            {
                var portfolios = new List<Portfolio>();
                try
                {
                    var conn = new SqlConnection(connectionString);
                    conn.Open();
                    var command = new SqlCommand("SELECT * FROM Portfolio ORDER BY
Name", conn);
                    command.CommandType = CommandType.Text;
                    var reader = command.ExecuteReader();
                    if (reader.HasRows)
                    {
                        while (reader.Read()) // построчно считываем данные
                        {
                            portfolios.Add(new Portfolio(reader));
```

```
                }
            }
            reader.Close();
            command.Dispose();
            conn.Close();
        }
        catch (Exception e)
        {
            //TODO
        }
        return portfolios;
    }

    public void SetCriterionOnProperty(Criterion criterion, SqlConnection
conn, SqlTransaction transaction)
    {

        var command = new SqlCommand(
            string.Format("UPDATE Property SET CriterionId = {0} WHERE Id =
{1}", criterion.Id, criterion.PropertyId)
            , conn, transaction);
        command.CommandType = CommandType.Text;
        command.ExecuteNonQuery();
        command.Dispose();
    }

    public void UpdateCreditClass(int creditId, int classId, SqlConnection
conn, SqlTransaction transaction)
    {

        var command = new SqlCommand(
            string.Format("UPDATE Credit SET ClassId = {0} WHERE Id = {1}",
classId, creditId)
            , conn, transaction);
        command.CommandType = CommandType.Text;
        command.ExecuteNonQuery();
        command.Dispose();
    }
```

```
            public void ClearCriterionFromPortfolio(int portfolioId, SqlConnection
conn, SqlTransaction transaction)
            {

                var command = new SqlCommand(
                    string.Format("UPDATE Property SET CriterionId = NULL WHERE
PortfolioId = {0}", portfolioId)
                    , conn, transaction);
                command.CommandType = CommandType.Text;
                command.ExecuteNonQuery();
                command.Dispose();
            }

            public void Analyze(List<Criterion> criterions, int portfolioId)
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var transaction = conn.BeginTransaction();
                try
                {
                    ClearCriterionFromPortfolio(portfolioId, conn, transaction);
                    criterions.ForEach(x => SetCriterionOnProperty(x, conn,
transaction));
                    LoanClassifier.Train();
                    var creditsJson = GetCreditDataToClassifyJson(portfolioId, conn,
transaction);
                    var fileName = FileHelper.ToTxt(creditsJson, portfolioId);
                    var creditClasses = LoanClassifier.Recognize(fileName);
                    var                          credits                          =
Newtonsoft.Json.JsonConvert.DeserializeObject<IEnumerable<JObject>>(creditsJson);
                    for (int i = 0; i < creditClasses.Count; i++)
                    {
                        UpdateCreditClass((int)
credits.ElementAt(i)["Credit_System_Id"], creditClasses[i], conn, transaction);
                    }
                    var command = new SqlCommand("PortfolioAnalyze", conn,
transaction);
                    command.CommandType = CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@portfolioId", portfolioId);
                    command.ExecuteNonQuery();
```

```
                transaction.Commit();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                transaction.Rollback();
            }
            conn.Close();
        }

        public List<Property> GetProperties(int portfolioId)
        {
            var properties = new List<Property>();
            try
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand(string.Format("SELECT * FROM
Property WHERE portfolioId = {0} ORDER BY Name", portfolioId), conn);
                command.CommandType = CommandType.Text;
                var reader = command.ExecuteReader();
                if (reader.HasRows)
                {
                    while (reader.Read()) // построчно считываем данные
                    {
                        properties.Add(new Property(reader));
                    }
                }
                reader.Close();
                command.Dispose();
                conn.Close();
            }
            catch (Exception e)
            {
                //TODO
            }
            return properties;
        }

        public string GetCreditDataJson(int portfolioId, int? rowCount = null)
```

```
        {
            string res = "";
            try
            {
                var conn = new SqlConnection(connectionString);
                conn.Open();
                var command = new SqlCommand("GetCreditData", conn);
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@portfolioId", portfolioId);
                command.Parameters.AddWithValue("@rowCount", rowCount);
                var reader = command.ExecuteReader();
                //reader.Read();
                int i = 0;
                while (reader.Read())
                {
                    res += Convert.ToString(reader[0]);
                }

                reader.Close();
                command.Dispose();
                conn.Close();
            }
            catch (Exception e)
            {
                //TODO
            }
            return res;
        }

        public string GetCreditDataToClassifyJson(int portfolioId, SqlConnection
conn, SqlTransaction transaction)
        {
            var res = "";
            var command = new SqlCommand("GetCreditDataToClassify", conn,
transaction);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@portfolioId", portfolioId);
            var reader = command.ExecuteReader();
            while (reader.Read())
            {
```

```
                    res += Convert.ToString(reader[0]);
                }
                reader.Close();
                command.ExecuteNonQuery();
                command.Dispose();
                return res;
            }
        }
    }
    using System;
    using CNTK;
    using System.IO;
    using System.Linq;
    using System.Collections.Generic;

    namespace CreditAnalisis.Execution
    {
        public enum Activation
        {
            None,
            ReLU,
            Sigmoid,
            Tanh
        }
        public static class TestCommon
        {
            public static string TestDataDirPrefix;
        }
        public class ClassifyHelper
        {
            public static Function Dense(Variable input, int outputDim,
DeviceDescriptor device,
                Activation activation = Activation.None, string outputName = "")
            {
                if (input.Shape.Rank != 1)
                {
                    //
                    int newDim = input.Shape.Dimensions.Aggregate((d1, d2) => d1 *
d2);
                    input = CNTKLib.Reshape(input, new int[] { newDim });
```

```
                }

                Function fullyConnected = FullyConnectedLinearLayer(input, outputDim,
device, outputName);
                switch (activation)
                {
                    default:
                    case Activation.None:
                        return fullyConnected;
                    case Activation.ReLU:
                        return CNTKLib.ReLU(fullyConnected);
                    case Activation.Sigmoid:
                        return CNTKLib.Sigmoid(fullyConnected);
                    case Activation.Tanh:
                        return CNTKLib.Tanh(fullyConnected);
                }
            }

            public static Function FullyConnectedLinearLayer(Variable input, int
outputDim, DeviceDescriptor device,
                string outputName = "")
            {
                System.Diagnostics.Debug.Assert(input.Shape.Rank == 1);
                int inputDim = input.Shape[0];

                int[] s = { outputDim, inputDim };
                var timesParam = new Parameter((NDShape)s, DataType.Float,
                    CNTKLib.GlorotUniformInitializer(
                        CNTKLib.DefaultParamInitScale,
                        CNTKLib.SentinelValueForInferParamInitRank,
                        CNTKLib.SentinelValueForInferParamInitRank, 1),
                    device, "timesParam");
                var timesFunction = CNTKLib.Times(timesParam, input, "times");

                int[] s2 = { outputDim };
                var plusParam = new Parameter(s2, 0.0f, device, "plusParam");
                return CNTKLib.Plus(plusParam, timesFunction, outputName);
            }

            public static float ValidateModelWithMinibatchSource(
```

```
                    string modelFile, MinibatchSource testMinibatchSource,
                    int[] imageDim, int numClasses, string featureInputName, string
labelInputName, string outputName,
                    DeviceDescriptor device, int maxCount = 10000)
                {
                    Function model = Function.Load(modelFile, device);
                    var imageInput = model.Arguments[0];
                    var labelOutput = model.Outputs.Single(o => o.Name == outputName);

                    var featureStreamInfo =
testMinibatchSource.StreamInfo(featureInputName);
                    var labelStreamInfo = testMinibatchSource.StreamInfo(labelInputName);

                    int batchSize = 500;
                    int miscountTotal = 0, totalCount = 0;
                    while (true)
                    {
                        var minibatchData =
testMinibatchSource.GetNextMinibatch((uint)batchSize, device);
                        if (minibatchData == null || minibatchData.Count == 0)
                            break;
                        totalCount +=
(int)minibatchData[featureStreamInfo].numberOfSamples;

                        // expected labels are in the minibatch data.
                        var labelData =
minibatchData[labelStreamInfo].data.GetDenseData<float>(labelOutput);
                        var expectedLabels = labelData.Select(l =>
l.IndexOf(l.Max())).ToList();

                        var inputDataMap = new Dictionary<Variable, Value>() {
                            { imageInput, minibatchData[featureStreamInfo].data }
                        };

                        var outputDataMap = new Dictionary<Variable, Value>() {
                            { labelOutput, null }
                        };

                        model.Evaluate(inputDataMap, outputDataMap, device);
```

```
                    var outputData =
outputDataMap[labelOutput].GetDenseData<float>(labelOutput);
                    var actualLabels = outputData.Select(l =>
l.IndexOf(l.Max())).ToList();

                    //Console.WriteLine(actualLabels);
                    //Console.WriteLine(expectedLabels);
                    /*Console.WriteLine("actualLabels");
                    actualLabels.ForEach(l => Console.WriteLine(l));
                    Console.WriteLine("expectedLabels");
                    expectedLabels.ForEach(l => Console.WriteLine(l));*/

                    int misMatches = actualLabels.Zip(expectedLabels, (a, b) =>
a.Equals(b) ? 0 : 1).Sum();

                    miscountTotal += misMatches;
                    Console.WriteLine($"Validating Model: Total Samples =
{totalCount}, Misclassify Count = {miscountTotal}");

                    if (totalCount > maxCount)
                        break;
                }

                float errorRate = 1.0F * miscountTotal / totalCount;
                Console.WriteLine($"Model Validation Error = {errorRate}");
                return errorRate;
            }

            public static List<int> RecognizeMinibatchSource(
                string modelFile, MinibatchSource testMinibatchSource,
                 int numClasses, string featureInputName, string labelInputName,
string outputName,
                DeviceDescriptor device)
            {
                var result = new List<int>();
                Function model = Function.Load(modelFile, device);
                var imageInput = model.Arguments[0];
                var labelOutput = model.Outputs.Single(o => o.Name == outputName);
                var featureStreamInfo =
testMinibatchSource.StreamInfo(featureInputName);
```

```
                var labelStreamInfo = testMinibatchSource.StreamInfo(labelInputName);
                int batchSize = 500;
                while (true)
                {
                    var minibatchData =
testMinibatchSource.GetNextMinibatch((uint)batchSize, device);
                    if (minibatchData == null || minibatchData.Count == 0)
                        break;
                    // expected labels are in the minibatch data.
                    var labelData =
minibatchData[labelStreamInfo].data.GetDenseData<float>(labelOutput);
                    var expectedLabels = labelData.Select(l =>
l.IndexOf(l.Max())).ToList();

                    var inputDataMap = new Dictionary<Variable, Value>() {
                        { imageInput, minibatchData[featureStreamInfo].data }
                    };
                    var outputDataMap = new Dictionary<Variable, Value>() {
                        { labelOutput, null }
                    };
                    model.Evaluate(inputDataMap, outputDataMap, device);
                    var outputData =
outputDataMap[labelOutput].GetDenseData<float>(labelOutput);
                    var actualLabels = outputData.Select(l => l.IndexOf(l.Max()) +
1).ToList();
                    result.AddRange(actualLabels);

                }
                return result;//Chess.getChessmanByNumber(actualLabels[0]).Name;
            }

            public static void SaveAndReloadModel(ref Function function,
IList<Variable> variables, DeviceDescriptor device, uint rank = 0)
            {
                string tempModelPath = "feedForward.net" + rank;
                File.Delete(tempModelPath);

                IDictionary<string, Variable> inputVarUids = new Dictionary<string,
Variable>();
```

```
                IDictionary<string, Variable> outputVarNames = new Dictionary<string,
Variable>();

                foreach (var variable in variables)
                {
                    if (variable.IsOutput)
                        outputVarNames.Add(variable.Owner.Name, variable);
                    else
                        inputVarUids.Add(variable.Uid, variable);
                }

                function.Save(tempModelPath);
                function = Function.Load(tempModelPath, device);

                File.Delete(tempModelPath);

                var inputs = function.Inputs;
                foreach (var inputVarInfo in inputVarUids.ToList())
                {
                    var newInputVar = inputs.First(v => v.Uid == inputVarInfo.Key);
                    inputVarUids[inputVarInfo.Key] = newInputVar;
                }

                var outputs = function.Outputs;
                foreach (var outputVarInfo in outputVarNames.ToList())
                {
                    var newOutputVar = outputs.First(v => v.Owner.Name ==
outputVarInfo.Key);
                    outputVarNames[outputVarInfo.Key] = newOutputVar;
                }
            }

            public static bool MiniBatchDataIsSweepEnd(ICollection<MinibatchData>
minibatchValues)
            {
                return minibatchValues.Any(a => a.sweepEnd);
            }

            public static void PrintTrainingProgress(Trainer trainer, int
minibatchIdx, int outputFrequencyInMinibatches)
```

```
            {
                if ((minibatchIdx % outputFrequencyInMinibatches) == 0 &&
trainer.PreviousMinibatchSampleCount() != 0)
                {
                    float trainLossValue =
(float)trainer.PreviousMinibatchLossAverage();
                    float evaluationValue =
(float)trainer.PreviousMinibatchEvaluationAverage();
                    Console.WriteLine($"Minibatch: {minibatchIdx} CrossEntropyLoss =
{trainLossValue}, EvaluationCriterion = {evaluationValue}");
                }
            }

            public static void PrintOutputDims(Function function, string
functionName)
            {
                NDShape shape = function.Output.Shape;

                if (shape.Rank == 3)
                {
                    Console.WriteLine($"{functionName} dim0: {shape[0]}, dim1:
{shape[1]}, dim2: {shape[2]}");
                }
                else
                {
                    Console.WriteLine($"{functionName} dim0: {shape[0]}");
                }
            }

        }
    }
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Text;
    using NPOI;
    using NPOI.XSSF.UserModel;
    using NPOI.SS.UserModel;
    using Newtonsoft.Json.Linq;
    using System.Data;
```

```
using Newtonsoft.Json;
using System.Linq;

namespace CreditAnalisis.Execution
{
    public class FileHelper
    {
        public static string DataFolder = Directory.GetCurrentDirectory() +
"\\Resources";
        public static string FromExcelToString(string filename)
        {
            XSSFWorkbook wb;
            string result = "";
            using (FileStream file = new FileStream(filename, FileMode.Open,
FileAccess.Read))
            {
                wb = new XSSFWorkbook(file);
            }

            ISheet sheet = wb.GetSheetAt(0);
            for (int rowNum = 1; rowNum <= sheet.LastRowNum; rowNum++)
            {
                IRow row = sheet.GetRow(rowNum);
                if (row != null)
                {
                    foreach (ICell cell in row.Cells)
                    {
                        if (cell.ColumnIndex == 0)
                            result += "|labels " + string.Join(' ',
Loan.getLoanClassById((int)cell.NumericCellValue).Label) + " |features ";
                        else
                            result += cell.NumericCellValue + " ";
                    }
                    result += Environment.NewLine;
                }
            }
            return result;
        }

        public static string FromCsvToTxt(string filename, int? patternId)
```

```
            {
                var k = 0;
                var result = "";
                try
                {
                    string data = "";
                    using (var stream = new FileStream(filename, FileMode.Open,
FileAccess.Read))
                    using (var reader = new StreamReader(stream))
                    {
                        data = reader.ReadToEnd();
                    }
                    var dataList = data.Split(Environment.NewLine).ToList();
                    dataList.RemoveAt(0);

                    foreach (var row in dataList)
                    {
                        var i = 0;
                        k++;
                        int c;
                        foreach(var cell in row.Split(";"))
                        {
                            if (k == 427)
                            {
                                System.Console.Write("");
                            }
                            if (cell != null && int.TryParse(cell, out c)) {
                                if (i == 0)
                                    result += "|labels " + string.Join(' ',
Loan.getLoanClassById(int.Parse(cell)).Label) + " |features ";
                                else
                                    result += int.Parse(cell) + " ";
                            }
                            i++;
                        }
                        result += Environment.NewLine;
                    }
                } catch (Exception e)
                {
                    System.Console.WriteLine(k);
```

```
                }
                var modelPath = Path.Combine(DataFolder,
string.Format("Model{0}.txt", patternId));
                File.WriteAllText(modelPath,
result.TrimEnd(Environment.NewLine.ToCharArray()));
                return modelPath;
            }

            public static List<int> FromCsv(string filename)
            {
                var result = new List<int>();
                string data = "";
                using (var stream = new FileStream(filename, FileMode.Open,
FileAccess.Read))
                using (var reader = new StreamReader(stream))
                {
                    data = reader.ReadToEnd();
                }
                var dataList = data.Split(Environment.NewLine).ToList();
                dataList.RemoveAt(0);
                var i = 0;
                foreach (var row in dataList)
                {
                    if (i == 10000)
                        break;
                    result.Add(int.Parse(row.Split(";")[0]));
                    i++;
                }
                return result;
            }

            public static string ToTxt(string json, int id)
            {
                //var result = "";
                DataSet dataSet = JsonConvert.DeserializeObject<DataSet>("{'Table1':
" + json + "}");
                DataTable dataTable = dataSet.Tables["Table1"];
                var classNumber = dataTable.Columns.Count - 1;
                dataTable.Columns.RemoveAt(0);
                var valueLines = dataTable.AsEnumerable()
```

```
                                    .Select(row => "|labels 1" + " |features " +
string.Join(" ", row.ItemArray));
                    //lines.AddRange(valueLines);
                    //result = string.Join(Environment.NewLine, valueLines);
                    //return result;

                    var filePath = Path.GetTempPath();
                    var filename = Path.Combine(filePath,
string.Format("PortfolioAnalyze_{0}.txt", id));
                    File.WriteAllLines(filename, valueLines.SkipLast(1));
                    return filename;
                }


                public static string[][] ExcelToArray(string filename)
                {
                    return new string[1][];
                }

                public static string Export(string json)
                {
                    DataSet dataSet = JsonConvert.DeserializeObject<DataSet>("{'Table1':
" + json + "}");
                    DataTable dataTable = dataSet.Tables["Table1"];

                    //var lines = new List<string>();
                    string[] columnNames = dataTable.Columns.Cast<DataColumn>().
                                                    Select(column =>
column.ColumnName).
                                                    ToArray();

                    var lines = string.Join(";", columnNames) + Environment.NewLine;
                    //lines.Add(header);

                    var valueLines = dataTable.AsEnumerable()
                                    .Select(row => string.Join(";", row.ItemArray));
                    //lines.AddRange(valueLines);
                    lines += string.Join(Environment.NewLine, valueLines);
                    return lines;
                    //File.WriteAllLines(@"C:\Users\PC\Downloads\excel.csv", lines);
```

```
        }

    }
}
using System;
using System.Collections.Generic;
using System.Text;

namespace CreditAnalisis.Execution
{
    class Loan
    {
        private static List<LoanClass> LoanClasses = new List<LoanClass>() {
            new LoanClass("Standart", 1, new byte[] { 1, 0, 0, 0, 0 }),
            new LoanClass("Not standart", 2, new byte[] { 0, 1, 0, 0, 0 }),
            new LoanClass("Doubtful", 3, new byte[] { 0, 0, 1, 0, 0 }),
            new LoanClass("Problem", 4, new byte[] { 0, 0, 0, 1, 0 }),
            new LoanClass("Hopeless", 5, new byte[] { 0, 0, 0, 0, 1 })
        };

        public static LoanClass getLoanClassByName(string name)
        {
            return LoanClasses.Find(x => x.Name == name);
        }

        public static LoanClass getLoanClassById(int id)
        {
            return LoanClasses.Find(x => x.Id == id);
        }

        public static LoanClass getLoanClassByLabel(byte[] label)
        {
            return LoanClasses.Find(x => x.Label.Equals(label));

        }
    }

    class LoanClass
    {
        public string Name;
```

```
            public byte[] Label;
            public int Id;

            public LoanClass(string name, int id, byte[] label)
            {
                Name = name;
                Id = id;
                Label = label;
            }
        }
    }
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using CNTK;
    using System.IO;

    namespace CreditAnalisis.Execution
    {
        public class LoanClassifier
        {
            public static string DataFolder =
@"C:\Users\PC\source\repos\CreditAnalisis\CreditAnalisis.Web\Resources";
            public static string TestDataFolder =
Directory.GetParent(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullNam
e).FullName + "\\Resources\\test";
            public static string featureStreamName = "features";
            public static int numClasses = 5;
            public static string labelsStreamName = "labels";
            public static string modelFile = "MNISTMLP.model";
            public static DeviceDescriptor device = DeviceDescriptor.CPUDevice;
            public static string classifierName = "classifierOutput";
            public static int[] dim = new int[] { 12 };
            //public static Image img = getImage();

            public static string Train(string fileName = null, bool forceRetrain =
false, int epochs = 10, double rate = 0.0001, Activation activation =
Activation.Sigmoid)
```

```
            {
                IList<StreamConfiguration> streamConfigurations =
getStreamConfigurations();
                Function classifierOutput;
                fileName = fileName ??
Path.Combine(@"C:\Users\PC\source\repos\CreditAnalisis\CreditAnalisis.Web\Resources",
"StandartTrain.txt");

                //If a model already exists and not set to force retrain, validate
the model and return.
                //if (File.Exists(modelFile) && !forceRetrain)
                //{
                //    //var minibatchSourceExistModel =
MinibatchSource.TextFormatMinibatchSource(
                //    //    Path.Combine(DataFolder, "Train.txt"),
streamConfigurations);
                //    //ClassifyHelper.ValidateModelWithMinibatchSource(modelFile,
minibatchSourceExistModel,
                //    //                    dim, numClasses, featureStreamName,
labelsStreamName, classifierName, device);
                //    return null;
                //}

                // build the network
                var input = CNTKLib.InputVariable(dim, DataType.Float,
featureStreamName);

                // For MLP, we like to have the middle layer to have certain amount
of states.
                int hiddenLayerDim = 200;
                var scaledInput =
CNTKLib.ElementTimes(Constant.Scalar<float>(0.00390625f, device), input);
                classifierOutput = CreateMLPClassifier(device, numClasses,
hiddenLayerDim, scaledInput, classifierName, activation);


                var labels = CNTKLib.InputVariable(new int[] { numClasses },
DataType.Float, labelsStreamName);
                var trainingLoss = CNTKLib.CrossEntropyWithSoftmax(new
Variable(classifierOutput), labels, "lossFunction");
```

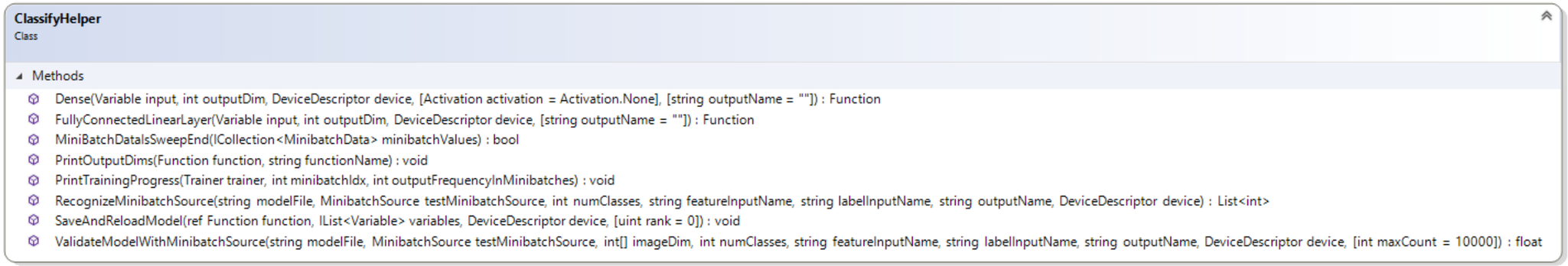
```
                var prediction = CNTKLib.ClassificationError(new
Variable(classifierOutput), labels, "classificationError");

                //ToTxt();
                // prepare training data
                var minibatchSource =
MinibatchSource.TextFormatMinibatchSource(fileName, streamConfigurations,
MinibatchSource.InfinitelyRepeat);

                var featureStreamInfo =
minibatchSource.StreamInfo(featureStreamName);
                var labelStreamInfo = minibatchSource.StreamInfo(labelsStreamName);

                // set per sample learning rate
                CNTK.TrainingParameterScheduleDouble learningRatePerSample = new
CNTK.TrainingParameterScheduleDouble(
                    rate, 1);

                IList<Learner> parameterLearners = new List<Learner>() {
Learner.SGDLearner(classifierOutput.Parameters(), learningRatePerSample) };
                var trainer = Trainer.CreateTrainer(classifierOutput, trainingLoss,
prediction, parameterLearners);

                //
                const uint minibatchSize = 64;
                int outputFrequencyInMinibatches = 500, i = 0;
                //int epochs = 100;
                while (epochs > 0)
                {
                    var minibatchData =
minibatchSource.GetNextMinibatch(minibatchSize, device);
                    var arguments = new Dictionary<Variable, MinibatchData>
                    {
                        { input, minibatchData[featureStreamInfo] },
                        { labels, minibatchData[labelStreamInfo] }
                    };

                    trainer.TrainMinibatch(arguments, device);
                    ClassifyHelper.PrintTrainingProgress(trainer, i++,
outputFrequencyInMinibatches);
```

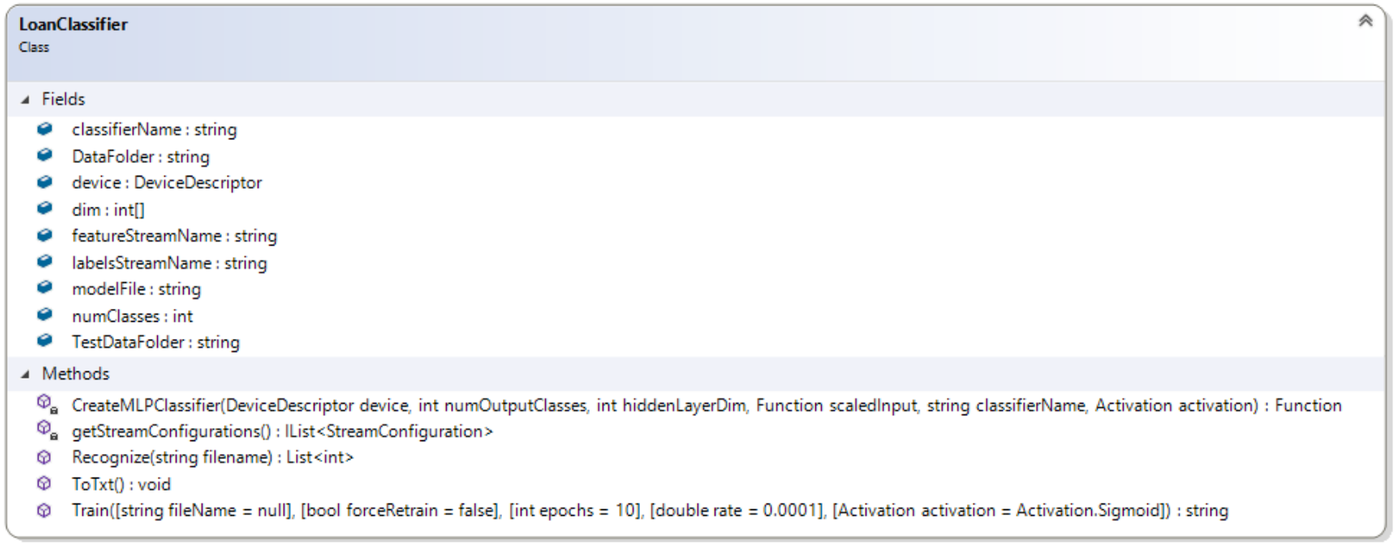
```
                    if (ClassifyHelper.MiniBatchDataIsSweepEnd(minibatchData.Values))
                    {
                        epochs--;
                    }
                }

                // save the trained model
                classifierOutput.Save(fileName == null ? modelFile :
fileName.Split('\\').Last().Split('.').First() + ".model");

                //var minibatchSourceExistModel =
MinibatchSource.TextFormatMinibatchSource(
                //        fileName, streamConfigurations);
                //ClassifyHelper.ValidateModelWithMinibatchSource(modelFile,
minibatchSourceExistModel,
                //                    dim, numClasses, featureStreamName,
labelsStreamName, classifierName, device);
                return "Done";
            }

            public static List<int> Recognize(string filename)
            {

                var minibatchSourceNewModel =
MinibatchSource.TextFormatMinibatchSource(filename, getStreamConfigurations(),
                    MinibatchSource.FullDataSweep);

                return ClassifyHelper.RecognizeMinibatchSource(modelFile,
minibatchSourceNewModel,
                     numClasses, featureStreamName, labelsStreamName, classifierName,
device);
            }

            private static IList<StreamConfiguration> getStreamConfigurations()
            {
                int dim = 12;

                return new StreamConfiguration[]
```

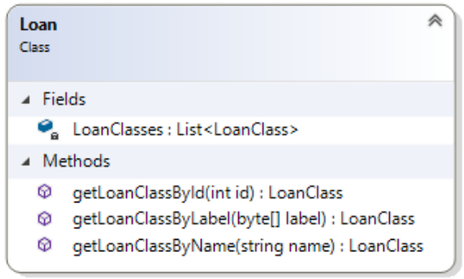
```
                    { new StreamConfiguration(featureStreamName, dim), new
StreamConfiguration(labelsStreamName, numClasses) };
            }

            private static Function CreateMLPClassifier(DeviceDescriptor device, int
numOutputClasses, int hiddenLayerDim,
                Function scaledInput, string classifierName, Activation activation)
            {
                Function dense1 = ClassifyHelper.Dense(scaledInput, hiddenLayerDim,
device, activation, "");
                Function classifierOutput = ClassifyHelper.Dense(dense1,
numOutputClasses, device, Activation.None, classifierName);
                return classifierOutput;
            }

            //public static void ToTxt()
            //{
            //    var train = FileHelper.FromExcelToString(Path.Combine(DataFolder,
"Dataset.xlsx"));
            //    File.WriteAllText(Path.Combine(DataFolder, "StandartTrain.txt"),
train);
            //    //var test = FileHelper.FromExcelToString(Path.Combine(DataFolder,
"Test.xlsx"));
            //    //File.WriteAllText(Path.Combine(DataFolder, "Test.txt"), test);
            //}

        }
    }
```

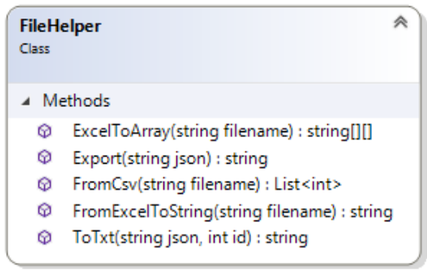
ДОДАТОК Б ГРАФІЧНИЙ МАТЕРІАЛ

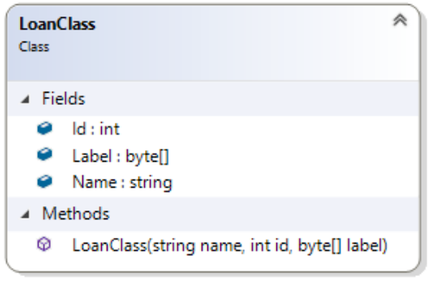
Схема структурна класів модулю Execution

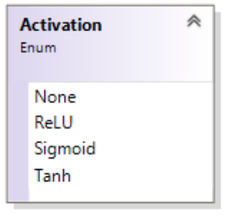












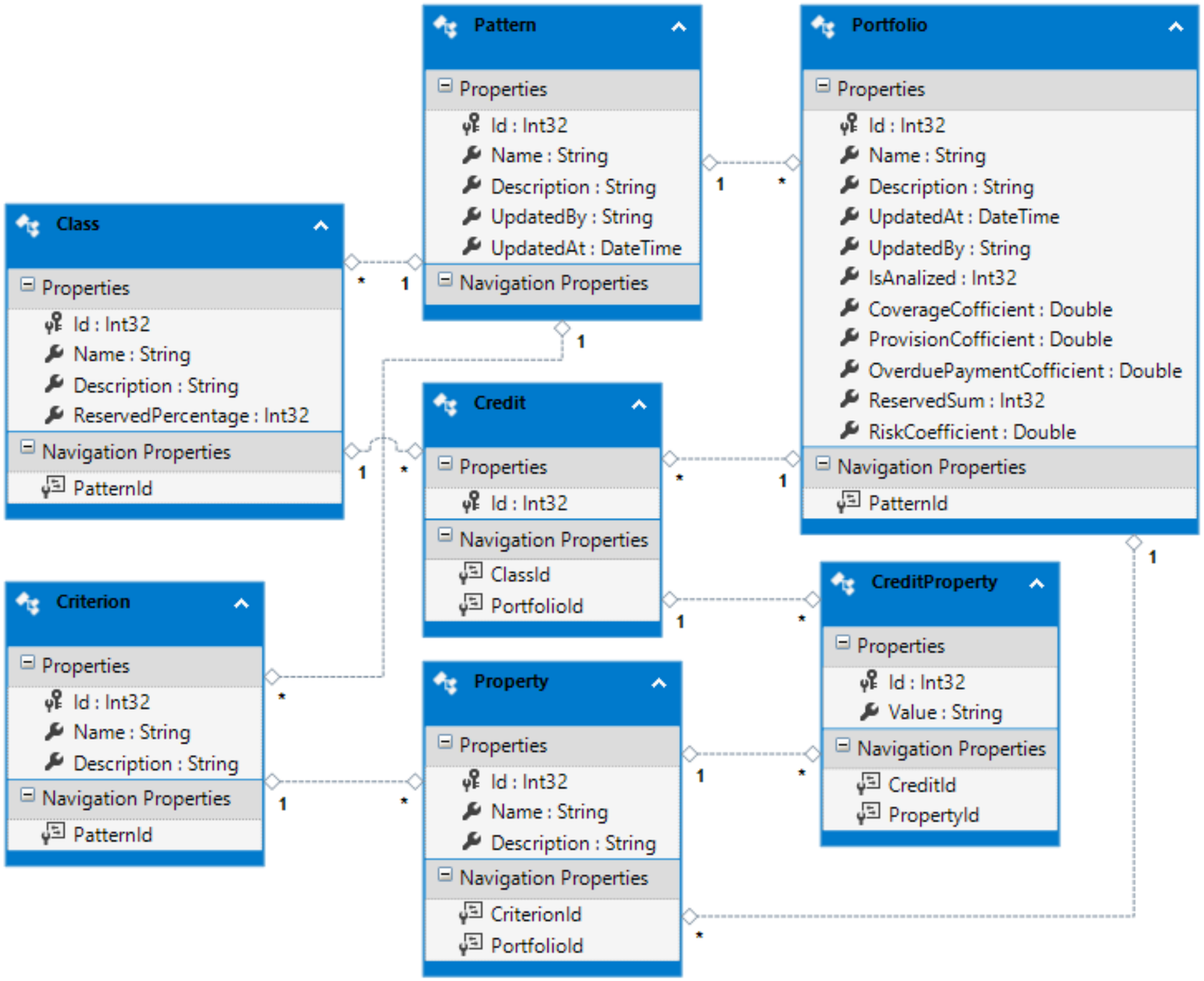
Демонстраційний плакат до магістерської дисертації

Інтелектуальна система для аналізу кредитного портфеля банка

Виконав студент гр. ІП-82мп Романовський Ю.О.

Керівник Селін Ю.М.

Діаграма сутностей



Демонстраційний плакат до магістерської дисертації

Інтелектуальна система для аналізу кредитного портфеля банка

Виконав студент гр. ІП-82мп — Романовський Ю.О.

Керівник — Селін Ю.М.